ZX SPECTRUM

(TS 2068)

Programación de juegos en lenguaje ensamblador

Stuart Nicholls





ZX SPECTRUM (TS 2068)

Programación de Juegos en lenguaje ensamblador

ZX SPECTRUM (TS 2068)

Programación de Juegos en lenguaje ensamblador

Stuart Nicholls

Traducción:

Gonzalo Olivares Ruiz
Departamento de Electrónica
Facultad de Ciencias
Universidad de Granada

María Jesús Vicente Pérez
Departamento de Inglés Científico
Universidad de Granada

Revisión técnica:

Antonio Vaquero Sánchez
Catedrático de Informática
Facultad de Ciencias Físicas
Universidad Complutense de Madrid

McGraw-Hill

MADRID • BOGOTÁ • BUENOS AIRES • GUATEMALA • LISBOA • MÉXICO
NUEVA YORK • PANAMÁ • SAN JUAN • SANTIAGO • SAO PAULO
AUCKLAND • HAMBURGO • JOHANNESBURGO • LONDRES • MONTREAL
NUEVA DELHI • PARÍS • SAN FRANCISCO • SINGAPUR • ST. LOUIS • SIDNEY
TOKIO • TORONTO

ZX SPECTRUM (TS 2068). Programación de Juegos en lenguaje ensamblador

Atlacomulco, 499-501, Naucalpan de Juárez, Edo. de México
Miembro de la Cámara Nacional de la Industria Editorial, Reg. Núm. 465

ISBN: 968-451-800-5

Traducido de la primera edición en inglés de
Assembly Language for Arcade Games and other fast Spectrum Programs

ISBN: 0-07-084729-0

Edición exclusiva para Ediciones La Colina, S. A. (España)

ISBN: 84-7615-065-2
Depósito legal: M. 31.227-1985

Compuesto en Grafilia, S. L. Pajaritos, 19. 28007 Madrid
Impreso en Gráficas EMA. Miguel Yuste, 27. 28037 Madrid

PRINTED IN SPAIN - IMPRESO EN ESPAÑA

A mi esposa Fran por la ayuda que me prestó pasando a máquina el manuscrito original (una tarea realizada sin entender ni una palabra del tema).

CONTENIDO

PREFACIO

¡Usted ya ha examinado con dificultad manuales sobre programación en lenguaje máquina Z80 para la Spectrum y se le ha dado un conocimiento claro de todas las instrucciones! Pero justo cuando empiezan a ponerse interesantes, terminan con unos cuantos ejemplos de cómo sumar y restar, y entonces debe usted apañárselas por sí mismo.

Este libro empieza donde terminan ese tipo de libros y le enseña a utilizar los conocimientos que ya tiene sobre las rutinas de la Spectrum, para hacer programas de trabajo completos, escritos totalmente en código máquina.

Estudiaremos las rutinas más fáciles, tales como la impresión, y también las rutinas más complicadas como las de números aleatorios y de cálculo con números de coma flotante.

El capítulo final le enseña a aplicar la información mostrada en el libro, para convertir completamente un programa BASIC de juegos en código máquina.

Los listados en código máquina han sido desarrollado con el Ensamblador de McGraw-Hill. No voy a insistir demasiado en el hecho de que cualquiera que se tome en serio la escritura de programas en lenguaje máquina debe usar un Ensamblador, ya que de esa forma se evitará la búsqueda tediosa en tablas, asociada normalmente con el lenguaje máquina, y lo que es más importante, le ayudará a modificar fácilmente su programa cuando se descontrole (lo que sucede a menudo cuando se empieza).

1 IMPRIMIR CON COLORES

Aunque la impresión en código máquina es una de las tareas más aburridas y que más tiempo lleva, también es una de las más importantes. Un buen programa puede estropearse por una mala composición, o por una visualización con mensajes de entrada ambiguos, y, de igual forma, se puede mejorar un programa mediocre con una buena composición. Antes de desarrollar un programa se debe tener muy en cuenta la presentación visual.

La Spectrum tiene 21 gráficos definibles por el usuario (UDGs) para hacer un uso completo de ellos. Por ejemplo, si usted está desarrollando un juego del tipo invasores del espacio, no se detenga redefiniendo los invasores, bombas y disparos; utilice los caracteres que tiene para redefinir los números desde el cero al nueve, para presentar las puntuaciones con caracteres futuristas y para crear paisajes interesantes. Utilice los colores de forma sensata; no ponga juntos el *rojo* y el *magenta* (o el *amarillo* y el *blanco*); puede ser conveniente ejecutar su programa con un aparato en blanco y negro para comprobar si aún así se ve bien, ya que muchos propietarios de la Spectrum no usan un aparato en color.

Como cada programa requiere la impresión de información en una u otra forma, en este capítulo veremos diferentes métodos de convertir la instrucción PRINT del BASIC al código máquina.

La primera orden de un programa en BASIC se refiere normalmente a las asignaciones de las sentencias BORDER, PAPER e INK (colores principales) seguida de CLS para poner esos colores en pantalla.

El color del BORDER (margen) es el más fácil de elegir en código máquina, y sólo necesita cinco bytes. En primer lugar se carga el registro A con el "número del color" requerido y después se hace una llamada a la rutina 8859 d (229Bh) de la ROM. Esta rutina selecciona en la pantalla el margen solicitado

```
1Ø BORDER 5: REM verde              LLAMADA ESTABLECER MARGEN

org 2376Ø                           23762 CD 9B 22      call 8859
2376Ø 3E Ø5          ld a,5         23765 C9            ret
```

Programa 1.1

y almacena el valor del color en la variable BORDCR 23624 del sistema (de hecho el valor almacenado es ocho veces el valor del color) es decir, los bits 5-3. Los otros bits son utilizados para almacenar los parámetros INK, BRIGHT, y FLASH de la mitad inferior de la pantalla (las *líneas de entrada*).

El programa 1.1 muestra la versión en código máquina para asignación de BORDER 5 (verde).

Los colores principales para PAPER e INK se mantienen en la variable ATTR-P (23693) del sistema. Los bits 0-2 guardan los colores INK, y los bits 3-5 los colores PAPER. Los bits 6 y 7 restantes se utilizan para FLASH y BRIGHT. Para FLASH Ø el bit 7 se pone a cero; para FLASH 1 el bit 7 se pone a uno. BRIGHT Ø tiene el bit 6 a cero y BRIGHT 1 tiene el bit 6 a uno.

El programa 1.2 muestra la asignación de PAPER 3: INK 1: CLS y demuestra el hecho de que ATTR-P (23693) puede ponerse como IY + 83. El registro IY se utiliza como puntero para la variable del sistema, y contiene el valor 23610. Si desea poner "LD(23609), A" por ejemplo, esto se puede considerar como (IY − 1) o (IY + 255).

```
5 REM establecer colores de pantalla
permanentes
1Ø PAPER 3: INK 1: CLS

LD (ATTR-P), colores
23762 FD 36 53 19 ld (iy+83), 25

Llamada abrir canal 2

3766 3E Ø2          ld a,2
23768 CD Ø1 16      call 5633

llamada CLS

23771 CD 6B ØD      call 3435
23774 C9            ret
```

Programa 1.2

La orden CLS (CALL 3435) borra la pantalla y establece los parámetros de pantalla, como el CLS del BASIC, *pero* observará que antes de hacer la llamada a CLS hay una llamada a la rutina 5633 de la ROM. Esto es muy importante, y se usará muchas veces en todo este Capítulo. La llamada es necesaria para abrir el canal 2, permitiendo que la información vaya a la mitad superior de la pantalla, es decir, a las líneas desde la 0 hasta la 21. Una vez establecidos nuestros parámetros permanentes de pantalla, podemos entonces pasar realmente a imprimir en la pantalla.

El programa 1.3 muestra la forma de imprimir un carácter en la pantalla en la *siguiente* posición de impresión; esta será Ø,Ø, si se ha ejecutado una orden CLS. Observará de nuevo que el "canal 2" debe abrirse primero, y que el código del carácter a imprimir debe estar en el registro A.

Se llama a la rutina ROM para imprimir un carácter utilizando RST 16 d, que de hecho es una llamada a la rutina 5618 d. Esta rutina imprime el carácter del registro A, y avanza una posición de impresión.

```
REM imprimir un carácter usando
colores permanentes
20 PRINT "A":

org 23760

Abrir canal 2

23760 3E 02          ld a,2
23762 CD 01 16       call 5633

CARGAR EN AC. CODIGO CAR. A

23765 3E 41          ld a,65

LLAMADA IMPRIMIR

23767 D7             rst 16
23768 C9             ret
```

Programa 1.3

El programa 1.4 amplía lo anterior para permitir la impresión de una palabra "corta". De nuevo se abre el canal 2, y se carga sucesivamente el registro A con cada carácter hasta que se imprime la palabra completa.

```
10 REM Imprimir una serie de caracteres
20 PRINT "HOLA"

org 23760

Abrir canal 2

23760 3E 02          ld a,2
23762 CD 01 16       call 5633
23765 3E 4B          ld a,72
H
23767 D7             rst 16
23768 3E 45          ld a,69
E
23770 D7             rst 16
23771 3E 4C          ld a,76
L
23773 D7             rst 16
23774 3E 4C          ld a,76
L
23776 D7             rst 16
23777 3E 4F          ld a,79
O

23779 D7             rst 16
23780 C9             ret
```

Programa 1.4

Sin embargo puede ver que el método anterior supondría una pérdida de tiempo y de memoria si hubiese que imprimir una palabra extensa o frase, y sería mejor abordar el problema de forma diferente.

Como se demuestra en el programa 1.5, hay varias opciones posibles. La idea principal consiste en guardar el mensaje con DATA, y leer un carácter cada vez hasta que se imprima todo el contenido de DATA.

```
10 REM Imprimir una serie LARGA
20 Print "AYUDAME"


org 23760
23760 3E 02          ld a,2
23762 CD 01 16       call 5633
23765 11 E4 5C       ld de, DATA
23768 01 07 00       ld bc,7

Longitud de serie

Lazo
23771 78             ld a,b
23772 B1             or c
23773 0B             dec bc
23774 C8             ret z
23775 1A             ld a,(de)
23776 13             inc de
23777 D7             rst 16
23778 18 F7          jr Lazo
DATA
defb 72 69 76 80 32 77 69
```

Programa 1.5

En la *versión 1* el registro DE contiene la dirección del comienzo de DATA y es utilizado como puntero para cada carácter; el registro BC contiene la longitud de la serie, y se hace una llamada RST 16 para cada carácter sucesivo hasta que BC = 0.

```
org 3760
23760 3E 02          ld a,2
23762 CD 01 16       call 5633
23765 11 DF 5C       ld de, DATA
23768 01 07 00       ld bc,7

LLAMADA IMPRIMIR SERIE

23771 CD 3C 20       call 8252
23774 C9             ret
DATA
defb 72 69 76 80 32 77 69
```

La *versión 2* es igual que la versión 1 salvo en que se utiliza la rutina 8252 d de la ROM, que de hecho es una copia de la rutina de la versión 1, desde la dirección 23771 a la 23778.

```
org 23760
23760 3E 02          ld a,2
23762 CD 01 16       call 5633
23765 11 E5 5C       ld de DATA
Lazo
23768 1A             ld a, (de)
23769 CB 7F          bit 7,a
23771 20 04          jr nz, END
23773 D7             rst 16
23774 13             inc de
23775 18 F7          jr Loop
END
23777 CB BF          res 7,a
23779 D7             rst 16
23780 C9             ret
DATA
defb 72 69 76 80 32 77 197
```

La *versión 3* puede utilizarse para imprimir series de cualquier longitud, pero está limitada a los códigos de carácter del 0 al 127. El último carácter se utiliza como marca para indicar el final de la serie añadiendo 128 a su código, asignando el bit 7, y entonces, antes de imprimir un carácter se hace una prueba para ver si el bit 7 es cero. Si lo es, se repite el lazo PRINT; si no, se hace un salto a la rutina END que pone a cero el bit 7,A, imprime el carácter, y continúa con el programa, en este caso con RET.

La *versión 3* puede utilizarse como "subrutina PRINT" en un programa en el que pueden almacenarse juntos varios mensajes

con indicadores de final, y todo lo que se necesitaría sería cargar DE con la dirección de comienzo del mensaje requerido y hacer una llamada a la subrutina PRINT. Con la Spectrum se utiliza una rutina similar, aunque más complicada, para imprimir palabras clave y mensajes de error; esta rutina empieza en la dirección 3082 d.

Los mensajes de error desde 1 a R, según se muestra en el Apéndice B del manual de la Spectrum, pueden imprimirse fácilmente utilizando la orden RST 8. El byte que sigue a la instrucción RST 8 apunta al mensaje de error requerido. Por ejemplo, si selecciona el mensaje "K invalid colour" que es de hecho el vigésimo-primer mensaje de error, utilice entonces:

```
RST 8
defb 19 d (número de mensaje – 2)
```

No se necesita ninguna instrucción de retorno, ya que la ROM volverá automáticamente al BASIC después de imprimir el error.

Ahora podemos movernos o imprimir caracteres o series en una línea y columna específica, como en la instrucción PRINT AT línea, columna; "mensaje" del BASIC. De nuevo, si mira en el Apéndice A del manual de la Spectrum, verá que los códigos de carácter del 6 al 23 son códigos de control, y en particular el código 22 es el código de "AT". Ahora, afortunadamente para nosotros, la instrucción RST 16 reconocerá el código 22 como "PRIM AT" y utilizará los dos bytes siguientes de DATA como x,y; estas son las coordenadas que nos dan la posición del mensaje. El programa 1.6 muestra esto usando defb 22,3,5 para PRINT AT 3,5;

```
10 REM Imprimir serie en x,y
20 PRINT AT 3,5; "Gracias"

org 23760
23760 3E 02          ld a,2
23762 CD 01 16       call 5633
23765 11 DF 5C       ld de,DATA
23768 01 0C 00       ld bc,12
23771 CD 3C 20       call 8252
23774 C9             ret
DATA

En 3 5

defb 22 3 5
defs Gracias
```

Programa 1.6

Puede utilizarse el mismo principio para TAB (CODE 23). Observe también que ENTER (CODE 13d) hará el mismo efecto que NEWLINE.

De modo que ahora podremos imprimir cualquier mensaje en cualquier posición de la línea 0 a la 21 utilizando los colores

principales. Todo lo que falta para completar la impresión de la pantalla normal es imprimir mensajes en colores poco comunes, BRIGHT, FLASH, OVER, e INVERSE.

El programa 1.7 muestra como se logra esto, y sigue los mismos principios del programa 1.6, en cuanto a que los códigos de control para PAPER, INK, BRIGHT, etc., se guardan en DATA con el byte que contiene el código de control siguiendo a la especificación de parámetros.

```
1000 REM Impresión en colores
1010 PRINT AT 4,7; PAPER 2; INK
4; FLASH 1; BRIGHT 1; OVER 1; IN
VERSE 1; "Puede leer esto"

org 23760
23760 3E 02          ld a,2
23762 CD 01 16       call 5633
23765 11 E2 5C       ld de, DATA
23768 01 24 00       ld bc,36
23771 CD 3C 20       call 8252

REINICIALIZAR COLORES PERMAMENTES
Y BRIGHT 0 FLASH 0

23774 CD 4D 0D       call 3405
23777 C9             ret
DATA
defb 22 4 7 17 2 16 4 18 1
defb 19 1 21 1 20 1
defs Puede leer esto
defb 21 0 20 0
```

Programa 1.7

Después de utilizar colores secundarios es necesario poner a cero los colores principales para seguir imprimiendo. En el caso de OVER y de INVERSE se requiere utilizar RST 16, pero PAPER, INK, BRIGHT y FLASH pueden reinicializarse mediante una llamada a la rutina 3405 d de ROM que copia el valor de ATTR P (23693) en ATTR T (23695).

Finalmente, consideramos la impresión en la mitad inferior de la pantalla, es decir en las líneas de entrada 22 y 23. Esta utiliza exactamente el mismo método que el que se describió para imprimir en la parte superior, excepto que hay que abrir un canal diferente antes de utilizar RST 16. El programa 1.8 muestra un ejemplo de como se lleva esto a cabo.

```
10 REM Imprimir en líneas INPUT
20 PRINT #0 AT 0,10; "Línea input
0";AT 1,10; "Línea input 1"
30 PAUSE 0

org 23760

Abrir canal -3

23760 3E FD          ld a,253
23762 DC 01 16       call 5633
23765 11 EA 5C       ld de, DATA
23768 01 1E 00       ld bc, 30
23771 CD 3C 20       call 8252

ESPERAR
23774 76             parar

Si pulsa tecla entonces
BIT 5, FLAGS será uno

23775 FD CB 01 6E bit 5, (iy+1)
23779 28 F9          jr z,WAIT
23781 FB CB 01 AE res 5, (iy+1)
23785 C9             ret
DATA

defb 22 0 10
defs Línea input 0
defb 22 1 10
defs Línea input 1
```

Programa 1.8

El canal en este caso es −3 (253).

El programa en BASIC es interesante porque en el manual no se menciona que con PRINT # Ø se imprimirá en las líneas de entrada. La instrucción PAUSE Ø es necesaria para parar el programa sin la instrucción Ø O.K. que podría cambiar la impresión de la línea de entrada.

De forma similar, esta rutina en código máquina requiere el equivalente a PAUSE Ø. Esto se logra examinando continuamente el bit 5 de FLAGS (23611) o (IY + 1) después de una instrucción HALT. La instrucción HALT espera a la siguiente rutina KEYSCAN, y si se pulsa una tecla, pone el bit 5 de FLAGS a uno. Así, esperamos en el lazo hasta que el bit 5 se pone a uno y entonces se vuelve al BASIC. Observar que la línea 22 se cuenta como línea Ø y la línea 23 como línea 1.

Una rutina ROM que puede ser muy útil para la impresión de las líneas de entrada, pero que deja intacta la parte superior de la pantalla, puede encontrarse en 3652, que es un "número específico de borrado de rutina de líneas"; el número de líneas se cuenta desde la parte inferior de la pantalla y debe ser mayor que 0. En el registro B se pone el número de líneas (1-24) y después se hace la llamada; los colores usados para borrar las líneas son los contenidos en ATTR P. Las líneas de entrada pueden borrarse con las instrucciones

```
LD B, 2
CALL 3652
(Return)
```

Antes de pasar al siguiente capítulo hay que destacar un punto importante referente a los gráficos definibles por el usuario. Si ha escrito un programa para ejecutarlo en máquinas de 16K y de 48K, entonces tendrá que situar correctamente los UDGs en cada máquina. La forma más fácil de hacer esto es situar los UDGs empezando a partir de la dirección 32600, es decir para una máquina de 16K, y después utilizar el Programa 1.9 para relocalizar los UDGs según la dirección guardada en la variable UDG (23675) del sistema.

```
  1Ø REM Tranferir 16 K de UDG's para
corregir localización para máquinas de 48K
  2Ø LET de=PEEK 23675+256*PEEK
23676
  3Ø LET h1=326ØØ
  4Ø LET bc=166
  5Ø POKE de, PEEK h1
  6Ø LET hl=hl+1
  7Ø LET de=de+1
  8Ø LET bc=bc−1
  9Ø IF bc<>Ø THEN GO TO 5Ø
 1ØØ REM volver
```

```
org 2376Ø
2376Ø ED 5B 7B 5C     ld de,(23675)
23764 21 58 7F        ld hl,326ØØ
23767 Ø1 A8 ØØ        ld bc,168
2377Ø ED BØ           ldir
23772 C9              ret
```

Programa 1.9

2 PLOT, DRAW, Y CIRCLE

PLOT

La orden PLOT x,y del BASIC lleva consigo la visualización del *pixel* columna x, fila y con un color primario. El pixel de la posición 0,0 está situado en la esquina inferior izquierda de la pantalla y el 175.255 en la esquina superior derecha.

En lenguaje máquina hay dos puntos de entrada útiles a la rutina PLOT de la ROM: el primero de ellos, que es probablemente el más útil y fácil de usar es CALL 8933. Antes de llamar a la rutina, el registro B debe contener el valor de y (en el rango de 0 a 175), y el registro C debe contener el valor de x (0-255). El programa 2.1 muestra la orden PLOT 75,125 del BASIC.

```
1000 REM Plot x,y
1010 PLOT 75,175

org 23760
23760 06 7D          ld b,125
23762 0E 4B          ld c,75
23764 CD E5 22       call 8933
23767 C9             ret
```

Programa 2.1

Si se usan números en decimal en vez de en hexadecimal, es más fácil seguir el programa cargando los registros B y C por separado. Por supuesto, podríamos haber ahorrado memoria usando LD BC,32075 pero esto sería confuso.

El segundo punto de entrada está en 8924. En este caso se requiere que los valores de x e y se coloquen en la *pila de la calculadora*, poniéndose el valor de y en el valor superior. El programa 2.2 demuestra esto con la misma PLOT 75,125.

```
org 23760
23760 3E 4B          ld a,75

STACK 75

23762 CD 28 2D       call 11560
23765 3E 7D          ld a,125

STACK 175

23767 CD 28 2D       call 11560

LLAMADA PLOT

23770 CD DC 22       call 8924
23773 C9             ret
```

Programa 2.2

La rutina 11560 de la ROM se utiliza para poner el valor del registro A en la parte superior de la pila de la calculadora. Con la rutina PLOT situada en la dirección 8924 se sacan los dos valores superiores de la pila y se traza un pixel. De esta forma la pila de la calculadora queda reducida en dos valores.

El uso de la calculadora se tratará más detalladamente en el programa 2.5. Este segundo punto de entrada es útil si la calculadora ha sido utilizada para manipular una fórmula para dibujar gráficos, etc.

El programa 2.3 muestra como se realiza PLOT OVER 1; x,y.

```
1Ø REM Plot OVER 1;x,y
2Ø PLOT OVER 1;25,4Ø


org 2376Ø
ESTABLECER OVER 1
ESTABLECER BITS Ø y 1 del FLAG P
2376Ø 3E Ø3          ld a,3
23762 FD 77 57       ld (iy+87),a
23765 Ø6 28          ld b,4Ø
23767 ØE 19          ld c,25
23769 CD E5 22       call 8933

ESTABLECER OVER Ø
ESTABLECER BITS Ø y 1 del FLAG-P
23772 AG             xor a
23773 FD 77 57       ld (iy+87),a
23776 C9             ret
```

Programa 2.3

Este programa pone a uno los bits 0 y 1 de P FLAG (IY + 87) antes de hacer la llamada PLOT, y después los pone a cero. Si no se pone a cero OVER Ø, entonces cualquier trazado o impresión futura se llevará a cabo como OVER 1.

PLOT INVERSE tiene la misma forma que PLOT OVER y requiere la puesta a uno de los bits 2 y 3 de P FLAG; la variable del sistema se pone al valor 12.

Usando el punto de entrada 8933 podemos escribir un programa para visualizar un carácter en cualquier lugar de la pantalla. La única limitación es que el código del carácter esté comprendido entre 32 (espacio) y 127 (copyright), es decir tiene que tener sus ocho-bytes de composición guardados en la ROM de generación de caracteres.

```
org 2376Ø
Cargar L con código de CAR

2376Ø 2E 7F          ld 1,127
23762 26 ØØ          ld h,Ø

Multiplicar ×8

23764 29             add hl,hl
23765 29             add hl,hl
23766 29             add hl,hl

Sumar a (236Ø6) para encontrar
comienzo de carácter

23767 ED 5B 36 5C 1d de,(236Ø6)
23771 19             add hl,de

Contador de byte

23772 Ø6 Ø8         ld b,8
L3                  Guardar punteo HL

23774 E5            push hl

Contador de bit

23775 ØE Ø8         ld c,8
23777 7E            ld a,(hl)
L2
23778 C5            push bc
```

```
Obtener posición x,y para plot

23779 ED 4B ØE 5D ld bc,(XY)

Examinar si BIT está a uno


23783 17                rla
23784 F5                push af



23785 3Ø Ø5             jr nc,L1


Si está a uno entonces PLOT

23787 C5                push bc
23788 CD E5 22          call 8933
23791 C                 pop bc
L1


Mover PLOT a x+1,y


23792 ØC                inc c
23793 ED 43 ØE 5D ld (XY),bc
23797 F1                pop af
23798 C1                pop bc
23799 ØD                dec c
238ØØ 2Ø E8             jr nz,12
238Ø2 C5                push bc


Reinicializar PLOT a x−8, y−1



238Ø3 ED 4B ØE 5D ld bc, (XY)
238Ø7 Ø5                dec b
238Ø8 3E F8             ld a,248
2381Ø 81                add a,c
23811 4F                ld c,a
23812 ED 43 ØE 5D       ld (XY),bc
23816 C1                pop bc
23817 E1                pop hl
23818 23                inc hl


Repetir hasta contador BYTE=Ø


23819 1Ø D1             djnz L3
23821 C9                ret
XY
defb 1ØØ 2Ø
```

Programa 2.4

En el programa 2.4 puede ver que el código del carácter requerido se coloca en el registro L (dirección 23761), después se multiplica por 8 y se suma con el valor almacenado en la variable del sistema (23606) para encontrar el comienzo en el generador de caracteres de los ocho bytes que componen el carácter. El registro HL se usa después como *puntero*, conteniendo la dirección del byte del carácter. El registro BC se usa como contador para la matriz de 8 bitx8bit del carácter, y el registro A contiene el byte del carácter.

La instrucción RLA se ejecuta 8 veces para cada byte de carácter, y si un bit se pone a uno, es decir, si después de RLA la bandera CARRY se pone a uno, entonces se visualiza el pixel.

Después de cada instrucción RLA se actualizan las coordenadas x,y. Las coordenada x,y iniciales son las del pixel superior izquierdo del carácter que está siendo visualizado. Por lo tanto, para que permanezca en la pantalla, el valor de y debe estar comprendido entre 7 y 175. El valor de x puede ser de 0 a 255; si se intenta tomar un valor de x superior a 255 resultará que el valor se hace 0, es decir, si C = 255 entonces “inc C” dará C = 0.

Este programa puede modificarse fácilmente para visualizar caracteres con una rotación de 90 grados en sentido contrario al de las agujas del reloj.

```
org 23760
Cargar L con código de CAR.

23760 2E 60          ld 1,96
23762 26 00          ld, h0

Multiplicar × 8

23764 29             add hl,hl
23765 29             add hl,hl
23766 29             add hl,hl

Sumar a (23606) para encontrar
comienzo de carácter

23767 ED 5B 36 5C    ld de, (23606)
23771 19             add hl,de

Contador de byte

23772 06 08          ld b,8
L3
Guardar puntero HL

23774 E5             push hl

Contador de bit

23775 0E 08          ld c,8
23777 7E             ld a(hl)
L2
23778 C5             push bc

Obtener posición x,y de plot

23779 ED 4B 0E 5D ld bc, (XY)

Examinar si BIT es uno

23783                rla
3784 F5              push af
23785 30 05          jr nc,L1

Si es uno entonces PLOT

23787 C5             push bc
23788 CD E5 22       call 8933
23791 C1             pop bc
L1

Mover PLOT a x,y+1

23792 04             inc b
23793 ED 43 0E 5D    ld (XY),bc
23797 F1             pop af
23798 C1             pop bc
23799 0D             dec C
23800 20 E8          jr nz,L2
23802C5              push bc

Reinicializar PLOT a x+1,y−8

23803 ED 4B 0E 5D    ld bc, (XY)
23807 0C             inc c
23808 3E FB          ld a,248
23810 80             add a,b
23811 47             ld b,a
23812 ED 42 0E 5D    ld (XY), bc
23816 C1             pop bc
23817 E1             pop hl
23818 23             inc hl

Repetir hasta que contador BITE=0

23819 10 D1          djnz L3
23821 C9             ret
XY
defb 100 20
```

Programa 2.5

En el programa 2.5 las coordenadas x,y se actualizan para visualizar un byte *vertical* en vez de uno *horizontal* como ocurría en el programa 2.4; ahora las coordenadas x,y indican el pixel inferior izquierdo, y entonces el rango para y va de 0 a 167.

Una vez estudiada la forma de visualizar un carácter podemos pasar ahora a nuestro primer programa serio para *visualizar un mensaje* en cualquier posición de la pantalla, y para que sea más útil, permitiremos la selección de su anchura y al-

tura. Se dispone de una copia del programa en cinta –LOAD "PLOTDEMO".

Para que sea útil como rutina de un programa en BASIC, la he enganchado a una rutina de "búsqueda" para que los parámetros altura/anchura, posiciones x,y de visualización, y mensaje, puedan asignarse utilizando variables del BASIC de forma que no haya necesidad de la orden POKE con códigos máquina. Si desea utilizar la rutina dentro de un programa en código máquina, la rutina de búsqueda es entonces innecesaria y puede omitirse; los parámetros pueden copiarse directamente en la memoria intermedia de la impresora. El programa BASIC es auto-explicativo y permite al lector experimentar con valores diferentes de altura/anchura, x,y y mensajes (a$).

El código máquina es una extensión del programa 2.4. Los valores de x,y,h y w se encuentran utilizando la rutina de búsqueda. Esta rutina trabaja con el almacén de variables hasta que encuentra la variable CODE requerida, y entonces almacena el valor de la variable (0-255) en la *memoria intermedia de la impresora*. Una vez que se ha almacenado el valor de la variable, se encuentra y se almacena LEN a $ de nuevo (0-255), y finalmente se copia el mensaje en la memoria intermedia de la impresora (el lugar más útil para almacenar datos). La rutina "PLOT" incluye una instrucción CATCHALL y convertirá cualquier parámetro equivocado en otro aceptable, y se compone de cinco bucles:

(L1) Dibujar un bit establecido con SET para WIDTHw
(L2) Repetir L1 para los ocho bits
(L3) Repetir HEIGHT h veces
(L4) Repetir para los ocho bits
(L5) Repetir hasta el final del mensaje

Observe el comentario del programa en las variables FOR/NEXT del lazo LOOP.

En el programa 2.6 el código máquina comienza en 32393. La primera rutina, de 32393 hasta 32418, inicializa el registro BC como puntero de la memoria intermedia de la impresora y llama a la rutina SET para establecer el código de la variable situada en la dirección 23728. La rutina SET llama sucesivamente a la rutina FIND para encontrar los códigos de las variables. Entonces el valor de la variable (0-255) se almacena en la dirección de la memoria intermedia de la impresora, indicada por el registro BC. Así, las variables x,y,h, y w se localizan y almacenan consecutivamente. La rutina de 32419 a 32442 encuentra el código de la variable a$, entonces se guarda la longitud (0-255)

INSTRUCCIONES

Este primer programa de demostración le permitirá «dibujar» (plot) mensajes en cualquier lugar, de la pantalla usando variables BASIC para establecer los parámetros del mensaje.

Los parámetros de caracteres se establecen como sigue:

Altura	:	LET h=1 to h=22
Anchura	:	LET h=1 to h=32
eje x	:	LET x=0 to 255
eje y	:	LET y=0 to 175
Mensaje	:	LET a$="*********"
Plot	:	RANDOMISE USR 32393

PULSE CUALQUIER TECLA

Puede seleccionar los colores de tinta antes de la llamada al código máquina PERO acuérdese de reinicializar después el color permanente.

Cualquier parámetro incorrecto será corregido con el código/M, y si el mensaje es demasiado largo/alto se enrroscará en la pantalla.

NOTA:

NO UTILIZAR LAS VARIABLES x,y,h EN LAZOS FOR/NEXT PUES EL PROGRAMA SE ESTROPEARA.
El siguiente programa le pedirá los parámetros y visualizará su mensaje.

PULSE Q PARA PARAR

```
   5 OVER Ø: PAPER 6: BORD ER 3: CLS
  1Ø LET x=24: LET y=79: LET =5: let
w=2: LET a$="PARAR LA CINTA"
  15 INK 2: RANDOMIZE USR 32393
  18 PAUSE 1ØØ
  2Ø LET w=8: LET y=150: LET x=0:
LET a$="PLOT"
  3Ø INK 1: RANDOMIZE USR 32393
  35 LET a$=____": LET y=140
  36 INK 1: RANDOMIZE USR 32393
  4Ø FLASH 1: GO SUB 8ØØØ
  5Ø FLASH Ø
  6Ø PAUSE Ø
  7Ø PAPER 7: INK Ø: BORDER 7: CLS
  8Ø LET: X=32: LET y:=175: LET h=1
: LET w=2: LET a$="INSTRUCCIONES"
  9Ø INK 2: RANDOMIZE USR 32393:
INK Ø
  1ØØ PRINT AT 2,0;"Este primer programa
de demostración le permitirá "dibujar" (plot)
mensajes en cualquier lugar de la pantalla
usando variables BASIC para establecer los
parámetros del mensaje».
  11Ø PRINT ''"Los parámetros de carac-
teres se establecen como sigue:–"
  12Ø PRINT '''Altura: LET h = 1 to
h=22"'"Anchura : LET w=1 to w=3 2"
  13Ø PRINT "eje x : LET x=Ø a 255" "eje y:
LET y=Ø a 175"
  14Ø PRINT "Mensaje : LET a$=";CHR$
34; "*********"CHR$ 34
  15Ø PRINT "Plot : RANDOMISE USR
32393"
  16Ø GO SUB 8ØØØ
  17Ø IF INKEY$<>"" THEN GO TO 17Ø
  175 IF INKEY$="" THEN GO TO 175 178
CLS
  18Ø PRINT: «Puede seleccionar los co-
lores de tinta antes de la llamada al código
máquina PERO acuérdese de reinicializar
después el color permanente.
  19Ø PRINT "Cualquier parámetro inco-
rrecto será corregido con el código /M, y si
el mensaje es demasiado largo/alto se en-
rroscará en la pantalla."
  2ØØ LET a$="NOTA": LET y=95: INK 2:
RAMDOMIZE USR 32393: INK Ø.
  21Ø PRINT AT 12,Ø; "NO UTILIZAR LAS
VARIABLES x,y,h EN LAZOS FOR/NEXT
PUES EL PROGRAMA SE ESTROPEARA"
  22Ø PRINT' "El siguiente programa le pe-
dirá los parámetros y visualizará su men-
saje."
  23Ø GO SUB 8ØØØ
  235 IF INKEY$<>"" THEN GO TO 235.
  24Ø IF INKEY$="" THEN GO TO 24Ø
  242 CLS : BORDER 4
  245 PLOT Ø,121: DRAW 255,Ø: DRAW
Ø,–73: DRAW –255,Ø: DRAW Ø,73
25Ø PRINT AT 8,Ø; "ALTURA 1–22 ? __",
"ANCHURA 1–32? __", "X
eje x Ø–255 ?       ___", "eje y  Ø-1
75 ? __MENSAJE ? ",," ___________________________
_________________TINTA
   Ø–9 ?»,,
  255 LET z$="
             "
  257 LET v$="
             "
  26Ø PRINT OVER 1;AT 8,Ø;z$
  262 INPUT h: PRINT AT ',18;h; " "
; OVER 1;AT 8,0;v$
  265 PRINT OVER 1;AT 9,Ø;z$: INPUT w:
PRINT AT 9,18; w; " "; OVER1 : AT 9,Ø; v$
  27Ø PRINT OVER 1;AT 1Ø,Øz$: INPUT x:
PRINT AT 1Ø,18; x; " "; OVER1; AT 1Ø,Ø;v$
  272 PRINT OVER 1; AT 11,Ø;z$: INPUT y:
PRINT AT 11,18; y; " "; OVER 1; AT 11,Ø; v$
  273 PRINT OVER 1;AT 12,Ø;z$;AT
13,Ø;z$: INPUT a$: PRINT AT 13,Ø; a$: FOR
k=LEN a$ TO 31: PRINT " ";: NEXT k:
PRINT OVER 1;AT 12,0; v$; OVER 1;AT
13,Ø; v$
  275 PRINT OVER 1;AT 14,Ø;z$: IN PUT i:
PRINT AT 14,Ø; INK i; OVER1; v$
  28Ø PAUSE 5Ø
  29Ø CLS : INK I: RANDOMIZE USR 3
2393: INK Ø
3ØØ PRINT #Ø;AT Ø,Ø; "PULSE Q PARA PA-
RAR"; AT 1,Ø; "CUALQUIER OTRA PARA IR
OTRA VEZ"
  31Ø PAUSE Ø: IF INKEY$="q" OR IN
KEY$="Q" THEN STOP
  32Ø CLS : GO TO 245
8ØØØ LET h=1: LET w=2: LET x=24:
LET y=15: LET a$="PULSE CUALQUIER
TECLA"
  INK 2: RANDOMIZE USR 32393: INK Ø
8Ø1Ø RETURN
9ØØØ STOP
98ØØ CLEAR 32334: LOAD ""CODE: GO
TO 5
99ØØ SAVE "PLOTDEMO" LINE 98ØØ
995Ø SAVE "LARGE" CODE 32335,265
org 2376Ø 32335
FIND
32335 2A 4B 5C          ld, hl, (23627)
L2
32338 3A BØ 5C          ld a,(23728)
32341 BE                cp (hl)
32342 CB                ret z
32343 CB 6E             bit 5,(hl)
32345 2Ø Ø8             jr nz,L1
32347 23                inc hl
32348 5E                ld e,(hl)
32349 23                inc hl
32350 56                ld d,(hl)
32351 19                add hl,de
32352 23                inc hl
32353 18 EF             jr L2
L1
32355 CB 76             bit 6,(hl)
32357 2Ø ØC             jr nz,L3
32359 23                inc hl

32360 7E                ld a,(hl)
32361 CB 7F             bit 7,a
32363 28 FA             jr z,–6
32365 11 Ø6 ØØ          ld de,6
32368 19                add hl,de
32369 18 DF             jr L2
```

```
L3
32371 CB 7E          bit 7,(hl)
32373 28 F6          jr z,−10
32375 11 13 ØØ       ld, de,19
32378 19             add hl,de
32379 18 D5          jr L2
SET
32381 32 BØ 5C       ld (23728),a
32384 CD 4F 7E       call FIND
32387 23             inc hl
32388 23             inc hl
32389 23             inc hl
32390 7E             ld a, (hl)
32391 Ø2             ld (bc),a
32392 C9             ret

EMPEZAR
32393 Ø1 ØØ 5B       ld bc,23296
32396 3E 78          ld a,12Ø
32398 CD 7D 7E       call SET
324Ø1 Ø3             inc bc
324Ø2 3E 79          ld a,121
324Ø4 CD 7D 7E       call SET
324Ø7 Ø3             inc bc
324Ø8 3E 68          ld a,1Ø4
3241Ø CD 7D 7E       call SET
32413 Ø3             inc bc
32414 3E 77          ld a,119
32416 CD 7D 7E       call SET
32419 3E 41          ld a,65
32421 32 BØ 5C       ld (23728),a
32424 CD 4F 7E       call FIND
32427 23             inc hl
32428 5E             ld d,(hl)
32429 23             inc hl
3243Ø 56             ld,(hl)
32431 ED 53 Ø4 5B    ld (233ØØ),de

32435 D5             push de
32436 C1             pop bc
32437 23             inc hl
32438 11 Ø5 5B       ld de, 233Ø1
32441 ED BØ          ldir
32443 2A ØØ 5B       ld hl, (23296)
32446 AF             xor a
32447 7C             ld a,h
32448 DE BØ          sbc a,176
3245Ø 38 Ø4          jr c,+4
32452 67             ld h,a
32453 22 ØØ 5B       ld (23296),hl
32456 22 BØ 5C       ld (23728),hl
32459 21 Ø5 5B       ld hl,233Ø1
LP5
32462 E5             push hl
32463 7E             ld a,(hl)
32464 26 ØØ          ld h,Ø
32466 6F             ld l,a
32467 29             add hl,hl
32468 29             add hl,hl
32469 29             add hl,hl
3247Ø 11 ØØ 3C       ld, de, 1536Ø
32473 19             add hl,de
32474 Ø6 Ø8          ld b,8
LP4
32476 C5             push bc
32477 ED 4B Ø1 5B    ld bc (23297)
LP3
32841 7E             ld a,(hl)
32482 E5             push hl
32483 C5             push bc
32484 Ø6 Ø8          ld b,8
LP2
32486 C5             push bc
32487 17             rla
32488 F5             push af
32489 DA F9 7E       jp c,PLOT
32492 2A Ø3 5B       ld hl,(23299)
32495 3A BØ 5C       ld a,(23728)
32498 85             add a,l
32499 32 BØ 5C       ld (23728),a

325Ø2 C3 ØF 7F       jp SKIP
PLOT
325Ø5 ED 4B BØ 5C    ld Bc,(23728)
LP1
325Ø9 C5             push bc
3251Ø ED 4B BØ 5C    ld bc,(23728)
32514 C5             push bc
32515 CD E5 22       call 8933
32518 C1             pop bc
32519 ØC             inc c
3252Ø ED 43 BØ 5C    ld (23728),bc
32524 C1             pop bc
32525 1Ø EE          djnz LP1
SKIP
32527 F1             pop af
32528 C1             pop bc
32529 1Ø D3          djnz LP2
32531 3A ØØ 5B       ld a,(23296)
32534 21 BØ 5C       ld hl,23728
32537 77             ld (hl),a
32538 23             inc hl

32539 AF             xor a
3254Ø 7E             ld a,(hl)
32541 DE BØ          sbc a,176
32543 38 Ø3          jr c,+3
32545 77             ld (hl),a
32546 18 Ø8          jr +8
32548 7E             ld a,(hl)
32549 FE ØØ          cp Ø
32551 2Ø Ø2          jr nz,+2
32553 36 BØ          ld (hl),176
32555 35             dec (hl)
32556 C1             pop bc
32557 E1             pop hl
32558 1Ø B1          djnz LP3
3256Ø 23             inc hl
32561 C1             pop bc
32562 1Ø AB          djnz LP4
32564 3A Ø3 5B       ld a,(23299)
32567 87             add a,a
32568 87             add a,a
32569 87             add a,a
3257Ø 6F             ld l,a
32571 3A BØ 5C       ld a,(23728)
32574 85             add a,l
32575 32 ØØ 5B       ld (23296),a
32578 32 BØ 5C       ld (23728), a
32581 3A Ø1 5B       ld a,(23297)
```

```
32584 32 B1 5C     ld (23729),a
32587 E1           pop hl
32588 23           inc hl
32589 3A Ø4 5B     ld a,(233ØØ)
32592 3D           dec a
32593 C8           ret z
32594 32 Ø4 5B     ld (233ØØ),a
32597 C3 CE 7E     jp LP5
```

Programa 2.6

y finalmente se copia la serie. La rutina de 32443 a 32458 examina el valor de y (si es mayor que 175 se cambia a y − 176) y después almacena los valores x e y anteriores en 23296/7 y también en 23728/9.

El código máquina restante es una extensión de la rutina plot (dibujar un carácter) con los bucles adicionales para visualizar con PLOT el pixel "w (anchura)" veces para "h líneas"; se hace una prueba después de cada "línea" para asegurarse de que la siguiente posición PLOT permanece en la pantalla. La rutina desde 32540 a 32554 comprueba la siguiente posición de PLOT y origina un efecto de enrosque si está fuera de la parte superior de la pantalla.

Sin profundizar ahora demasiado en las rutinas CALCULATOR (de la calculador) podemos estudiar un programa que hace uso de PLOT en el punto de entrada 8924.

El programa 2.7 muestra como puede dibujarse una onda sinuosidal utilizando CALCULATOR para manejar la fórmula:

88 + 80 * SEN (A/128 * PI)

```
1ØØØ FOR A=Ø TO 255
1Ø1Ø PLOT A,88+8Ø*SIN (A/128*PI)
1Ø2Ø NEXT A


org 2376Ø
LET A=Ø

2376Ø AF           xor a
LAZO
GUARDAR A
23761 F5           push af
APILAR A
23762 CD 28 2D     call 1156Ø
23765 3E 58        ld a,88
23767 CD 28 2D     call 1156Ø
2377Ø 3E 5Ø        ld a,8Ø
23772 CD 28 2D     call 1156Ø
23775 F1           pop af
23776 F5           push af
23777 CD 28 2D     call 1156Ø
2378Ø 3E 8Ø        ld a,128
23782 CD 28 2D     call 1156Ø

APILAR 88 8Ø A 128

USAR CALCULADORA

23785 EF           rst 4Ø
Dividir 2 valores superiores (A/128)
defb 5
Stack PI/2
deb 163
Duplicar valor superior
defb 49
Sumar 2 valores superiores (PI)
defb 15
Multiplicar 2 valores superiores
(A/128*PI)
defb 4
SEN valor superior SEN (A/128*PI)
defb 31
Mutplicar 2 valores superiores
8Ø*SIN (A/128*PI)
defb 4
Sumar 2 valores superiores
88+80*SIN (A/128*PI)

defb 15
Fin de rutina calculadora
Los 2 valores superiores son
A y 88+80*SIN (A/128*PI)
```

```
defb 56
LLAMADA PLOT
23795 CD DC 22        call 8924
Obtener valor A
23798 F1              pop af
23799 3C              inc a
Examinar si no cero (256)
23800 FE 00           cp 0
23802 20 D5           jr nz, LOOP
23804 C9              ret
```

Programa 2.7

La rutina CALCULATOR puede utilizarse para realizar operaciones aritméticas complejas mediante el uso de *literales*. En realidad son llamadas ROM para realizar cálculos en los cuales intervienen los dos valores situados en la parte superior de la pila de la calculadora, pero pueden usarse también para manejar series. Por ejemplo, si los dos valores de la parte superior fueran 1234 y 2, y el Literal 4 fuese llamado desde la rutina CALCULATOR, entonces esos dos valores podrían sacarse de la pila, multiplicarse y devolver su producto a la parte superior de la pila; no obstante la pila se reduciría en un valor, y el valor superior sería 2468.

Los números *enteros positivos* pueden colocarse en la pila por medio de:

STACK VAL A – CALL 11560 OR STACK VAL BC – CALL – 11563.

En el programa 2.7 he utilizado STACK VAL A para ordenar "variable A", 88,80 "variable A", y 128.

Después se llama a la rutina USE calculator mediante la instrucción RST 40. La rutina CALCULATOR opera con los bytes que siguen a RST 40 y actúa sobre cada uno, llamando a otras subrutinas según sea necesario. En nuestro programa el siguiente byte es 5, y Literal 5 significa "GOSUB DIVIDE". Entonces se toman los dos valores superiores de la pila, se divide el primero por el segundo, y se devuelve el resultado a la pila. El siguiente byte de la rutina es Literal 163; le indica a la calculadora "STACK PI/2" (PILA PI/2).

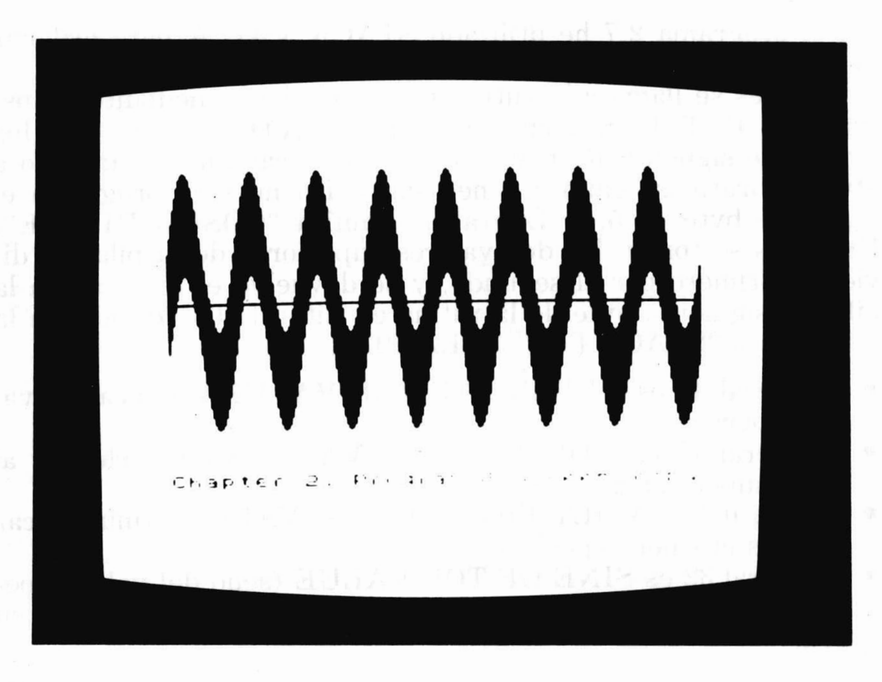
- El literal 49 es DUPLICATE TOP VALUE (duplicar el valor superior).
- El literal 15 es ADD TOP TWO VALUES (sumarle dos al valor superior) así se apila la suma.
- El literal 4 es MULTIPLY TOP TWO VALUES (multiplicar por dos el valor superior).
- El literal 32 es SINE OF TOP VALUE (seno del valor superior) de esta forma se reemplaza el valor superior por su seno.

Hay muchos más literales que pueden utilizarse en programas de juegos y serán tratados en capítulos posteriores.

Por último, el Literal 56 es “END CALCULATOR ROUTINE CALL” (llamada a la rutina fin de calculadora) que *debe* utilizarse para volver a su programa en código máquina. Otro punto importante a tener en cuenta cuando se utiliza la rutina CALCULATOR es que, antes de volver al BASIC deben borrarse todas las variables apiladas previamente.

Al final de la rutina CALCULATOR quedan dos valores en la pila de la calculadora; el valor superior es “88 + 80 * SEN (A/128 * PI)” y el otro es el valor de “A”. La llamada PLOT (8924) toma entonces esos dos valores, borra la pila, y dibuja (PLOT) el pixel. Después se incrementa el valor “A” y se repite LOOP hasta que “A” = 256 (es decir, “A” se pone a 0).

Con este programa puede ver que los literales de CALCULATOR son muy útiles y que una fórmula complicada se ha resuelto con sólo ocho bytes. En realidad necesita más bytes para utilizar la pila que para realizar los cálculos.



DRAW

Para la orden DRAW x,y hay también dos puntos de entrada. El primer punto (call 9402) requiere que el valor ABS de x esté en el registro C y que el valor ABS de y esté en el registro B. Se utiliza el registro DE para guardar el SGN de x e y; el registro D contiene SGNx (1 si es positivo, 255 si es negativo) y el registro E guarda SGNy. El programa 2.8 es una versión modificada del programa 2.7 para hacer una demostración de la orden DRAW.

```
 5 PLOT Ø,95: DRAW 255,Ø
1Ø FOR a=Ø TO 255
2Ø PLOT a,12Ø+4Ø*SIN (a/16*PI)
DRAW Ø,-5Ø
3Ø NEXT a


org 2376Ø
Plot Ø,95
2376Ø Ø6 5F          ld b,95
23762 ØE ØØ          ld c,Ø
23764 CD ¡E5, 22     call 8933
Guardar H'L'
23767 D9             exx
23768 E5             push hl
23769 D9             exx
Draw 255,Ø
2377Ø Ø6 ØØ          ld b,Ø
23772 ØE FF          ld c,255
23774 16 Ø1          ld d,1
23776 1E Ø1          ld e,1
23778 CD BA 24       call 94Ø2

Recuperar H'L'
23781 D9             exx
23782 E1             pop hl
23783 D9             exx
23784 AF             xor a

L1
23785 F5             push af
23786 CD 28 2D       call 1156Ø
23789 3E 78          ld a,12Ø
23791 CD 28 2D       call 1156Ø
23794 3E 28          ld a,4Ø
23796 CD 28 2D       call 1156Ø
23799 F1             pop af
238ØØ F5             push af
238Ø1 CD 28 2D       call 1156Ø
238Ø4 3E 1Ø          ld a,16
238Ø6 CD 28 2D       call 1156Ø
238Ø9 EF             rst 4Ø
defb 5 163 49 15 4 31 4 15 56

23189 CD DC 22       call 8924
Stoe H'L'
23822 D9             exx
23823 E5             push hl
23824 D9             exx
DRAW Ø,-5Ø
23825 Ø6 32          ld b,5Ø

23827 ØE ØØ          ld c,Ø
23829 16 FF          ld d,255
23831 1E Ø1          ld e,1
23833 CD BA 24       call 94Ø2
Restore H'L'
23836 D9             exx
23837 E1             pop hl
23838 D9             exx
23839 F1             pop af
2384Ø 3C             inc a
23841 FE ØØ          cp Ø
23843 2Ø C4          jr nz, L1
23845 C9             ret
```

Programa 2.8

Observe que antes de llamar a DRAW el registro HL debe ser almacenado (en el ejemplo se hace guardándolo en la pila) porque es utilizado en la rutina DRAW.

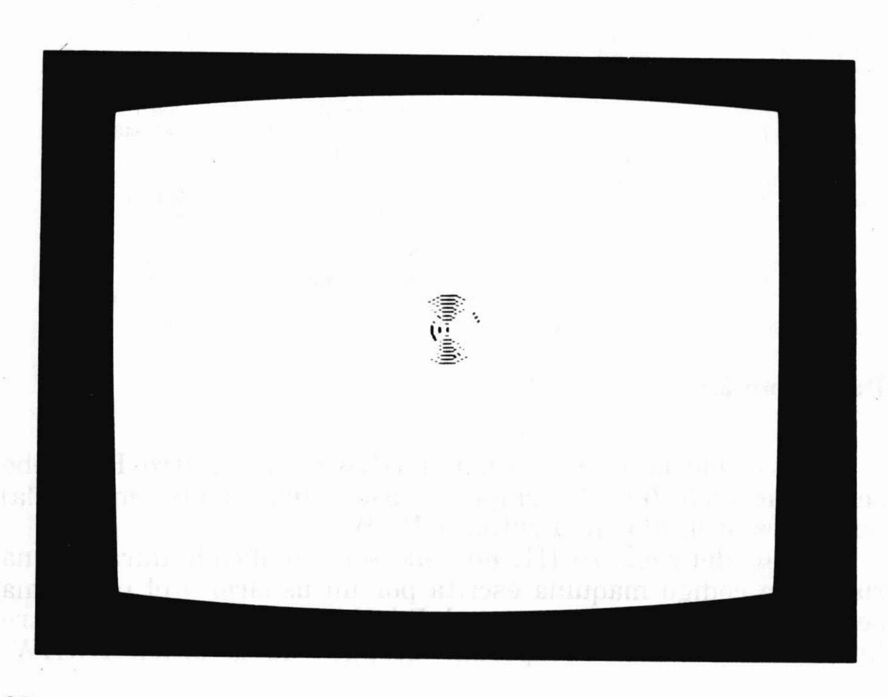
El valor del registro HL *no* debe ser modificado durante una rutina en código máquina escrita por un usuario o el programa se descontrolará al regresar al BASIC. El valor del registro fundamental HL es recuperado después de la orden DRAW.

DRAW OVER e INVERSE se utilizan de la misma forma que PLOT OVER/INVERSE.

El segundo punto de entrada para DRAW x,y requiere que los dos valores superiores de la pila de la calculadora contengan x e y; de nuevo, y es el valor superior –CALL 9335. Si desea utilizar este segundo punto de entrada necesitará leer el Capítulo 7 para tomar contacto con el método de la Spectrum de almacenar números *negativos* de cinco bytes en la pila de la calculadora.

DRAW x,y,a

El punto de entrada para esta orden BASIC es 9108 y requiere que los valores de x,y y a estén colocados en la pila de la calculadora (en ese mismo orden, es decir, "a" en la parte superior). De nuevo, en el Capítulo 7 se explicará como se apilan números negativos. Al igual que antes, tampoco debe cambiarse el valor del registro fundamental HL.



CIRCLE x,y,r

El punto de entrada para esta orden del BASIC es 9005 y requiere que los valores x,y, y r estén en la parte superior de la pila de la calculadora. El registro fundamental HL se guarda durante la llamada CIRCLE. El programa 2.9 muestra una forma de dibujar círculos concéntricos para conseguir un efecto de rizo.

```
10 REM Circle x,y,r
20 FOR B=1 TO 2
30 FOR A=1 TO 21 STEP 2
40 CIRCLE OVER 1,128,88,A
50 NEXT A
60 NEXT B;


org 23760
For B=1 TO 2

23760 06 02          ld, b2
L2
23762 05             push bc

For A=1 TO 21

23763 3E 01          ld a,1
L1
23765 F5             push af

Set OVER 1

23766 FD 36 57 03 ld (iy+87),3

Stack DATA

23770 3E 80          ld a,128

23772 CD 28 2D       call 11560
23775 3E 58          ld a,88
23777 CD 28 2D       call 11560
```

```
23780 F1             pop af
23781 F5             push af
23782 CD 28 2D       call 11560
Guardar H'L' (IMPORTANTE!)

23785 D9             exx
23786 E5             push hl
23787 D9             exx

Call CIRCLE

23788 CD 2D 23       call 9005

Recuperar H'L'

23791 D9             exx
23792 E1             pop hl
23793 D9             exx

23794 F1             pop af
STEP 2
23795 3C             inc a
23796 3C             inc a
23797 FE 17          cp 23
23799 20 DC          jr nz, L1
23801 C1             pop bc
23802 10 D6          djnz L2

Reinicializar OVER 0

23804 FD 36 57 00    ld (iy+87),0
23808 C9             ret
```

Programa 2.9

Observará que el programa es casi tan lento como el BASIC. Es porque la rutina CIRCLE de la ROM es bastante larga. Si su programa en código máquina necesita un círculo, es más rápido guardar los puntos del dibujo como DATA y usar un lazo para dibujar cada pixel.

3 CONTAR

La mayoría de los programas de juegos y muchos otros "serios" requieren alguna forma de contar. Esto se utiliza mucho en programas de juegos para llevar la puntuación, vidas perdidas, tiempo empleado, etc. Conozco tres métodos para contar y visualizar el resultado en la pantalla. El primero consume tanta memoria y es tan confuso, que sólo daremos una breve descripción. Este método lleva consigo la asignación del número MAX permitido con DATA. Por ejemplo, si usted estuviera contando desde 0 a 999999, entonces se reservarían seis bytes de DATA y se comenzaría por 000000. Después se necesitaría una rutina de impresión de series para imprimir los seis ceros en la pantalla. Cuando se incremente en uno el contador será necesario empezar por los bytes de las unidades, examinar que no es nueve, sumar uno a las unidades y después imprimir la serie. Si las unidades llegan al valor nueve se pondrían a cero, después la columna de las decenas se examina para ver si es nueve y así sucesivamente. Después de imprimir la serie se debe hacer una comprobación para ver si los seis bytes son iguales a nueve, y si es así se termina de contar. Si se quiere contar hacia adelante o hacia atrás en pasos mayores que uno hay que colocar el contador "incrementar en uno" en un lazo.

El segundo método se utiliza para contar hacia adelante o hacia atrás en un rango comprendido entre 0 y 65535 como se muestra en el programa 3.1.

```
org 2376Ø
Poner contador a cero

2376Ø Ø1 ØØ ØØ          1d bc,Ø
L1
23763 ED 43 BØ 5C       1d (23728), bc

Apilar valor

23767 CD 2B 2D          call 11563

Abrir canal 2

2377Ø 3 E Ø2            1d a, 2
23772 CD Ø1 16          call 5633

Establecer PRINT AT 11,16

23775 3E 16             ld a,22
23777 D7                rst 16

23778 3 E ØB            ld a,11
2378Ø D7                rst 16

23781 3E 1Ø             ld a,16
23783 D7                rst 16

Llamada IMPRIMIR VALOR EN PILA

23784 CD E3 2D          call 11747
```

```
Obtener valor y sumar 1
23787 ED 4B BØ 5C      ld bc, (23728)
23791 Ø3               inc bc

Volver si es cero
23792 78               ld a,b
23793 B1               or c
23794 C8               ret z

Examinar si pulsa tecla

23795 FD CB Ø1 6E bit 5, (iy+1)

23799 28 DA            jr z,L1
238Ø1 C9               ret
```

Programa 3.1

En este programa se utiliza una rutina ROM que imprime el valor que hay en la parte superior de la pila de la calculadora: call 11747. Los bytes que sobran en las variables 23728/9 del sistema son utilizados para guardar el valor del contador; el rango va desde 0 a 65535. Todo lo que necesitamos hacer después es coger el valor actual y sumarle uno, poner el nuevo valor otra vez en COUNTER (contador), apilar (STACK) el valor, hacer la asignación de parámetros con PRINT AT y llamar a PRINT VALUE ON STACK (imprimir el valor de la pila). Si el valor de BC se hace cero (es decir, 65.536) se vuelve al BASIC.

He incluido también una rutina EXIT situada en las direcciones desde 23795 a 23800 de forma que pulsando una tecla se volverá también al BASIC. Es aconsejable utilizarla, ya que de otra forma nos quedaríamos bloqueados en LOOP hasta que el contador llegue a 65535 y eso podría llevar varios minutos.

La tercera forma de contar utiliza la calculadora. Hasta ahora hemos considerado a la pila de la calculadora como un lugar para almacenar valores, sin explicar con detalle como se almacena. Los valores están realmente almacenados en la pila en forma de cinco-bytes; es decir, toman cinco bytes de información para definir cualquier número admitido por la computadora. En el Capítulo 7 se discute esto con más detalle. En el programa 3.2 hacemos uso de este hecho para contar desde 0 en adelante en pasos de 250 sin límite superior. Sin embargo, una vez que se llega a 99999999 se acude a la rutina PRINT STACK VALUE para imprimir el valor en su forma E (1E + 9 etc.), pero probablemente un valor de cien millones es suficientemente alto para la puntuación de cualquier juego.

```
org 2376Ø
Utilizar CALCULADORA

2376Ø EF               rst 4Ø

LITERAL STACK Ø
defb 16Ø

Terminar de usar CALCULADORA
defb 56

Establecer HL a STACKEND-5
después de instrucción defb 56

L1
Copiar valor en MEMORIA INTERMEDIA DE
IMPRESORA

23763 11 ØØ 5B         ld de,23296
23766 Ø1 Ø5 ØØ         ld bc,5
23769 ED BØ            ldir
```

```
Abrir canal 2

23771 3E Ø2           ld a,2
23773 CD Ø1 16        call 5633

Establecer PRINT AT 11,16
23776 3E 16           ld a,22
23778 D7              rst 16
23779 3E ØB           ld a,11
23781 D7              rst 16
23782 3E 1Ø           ld a,16
23784 D7              rst 16

Llamada PRINT VALUE ON STACK

23785 CD E3 2D        call 11747

STACK 25Ø

23788 3E FA           ld a,25Ø

2379Ø CD 28 2D        call 1156Ø

APILAR VALOR
23793 21 ØØ 5B        ld hl,23296
23796 ED 5B 65 5C     ld de,(23653)
238ØØ Ø1 Ø5 ØØ        ld bc,5
238Ø3 ED BØ           ldir
238Ø5 ED 53 65 5C     ld (23653),de

Usar CALCULADORA

238Ø9 EF              rst 4Ø

SUMAR LITERAL
defb 15
Terminar de usar CALCULADORA
defb 56

Examinar si se pulsa tecla

23812 FD CB Ø1 6E bit 5,(iy+1)

23816 28 C9           jr z,L1

Sacar último valor de PILA

23818 2A 65 5C        1d hl,(23653)
23821 2B              dec hl
23822 2B              dec hl
23823 2B              dec hl
23824 2B              dec hl
23825 2B              dec hl
23826 22 65 5C        ld (23653),hl
23829 C9              ret
```

Programa 3.2

El programa tiene varias rutinas importantes que requieren explicación. La asignación del valor 0 se realiza en la pila de la calculadora mediante defb 16Ø, que es el literal STACK Ø; los cinco bytes de la parte superior de la fila son después tomados y copiados en la memoria intermedia de la impresora para utilizarse como almacén. Esto se lleva a cabo con una instrucción LDIR; se asigna el valor de HL para nosotros con la rutina de la calculadora (STACKEND-5). El valor inicial se imprime en 11, 16 con la rutina PRINT VALUE y así se borra la pila.

Lo mismo con la rutina ADD 25Ø. Requiere que el valor de 250 esté colocado en la pila por medio de STACK VAL A, y después se pone en la pila el valor “almacenado”. Esto se realiza utilizando LDIR. El comienzo del valor almacenado se pone en el registro HL; el valor de STACKEND se coloca en DE. Después de la instrucción LDIR debe devolverse a la variable del sistema la nueva dirección de STACKEND. La calculadora se utiliza para sumar estos dos valores y para repetir las rutinas STORE y PRINT VALUE. De nuevo se añade una rutina de pulsación de tecla para permitir la salida de la cuenta, pero observe que la pila se reinicializará antes de volver al BASIC.

La cuenta hacia atrás requiere que el valor de comienzo se coloque en la pila (ver Capítulo 7 para números mayores que 65535) y que se use el Literal 3 SUBSTRACT. Observe que en la resta, el valor "superior" es sacado desde el inferior. Al final de cada cuenta se requiere una prueba para ver si los bytes del "almacén" están todos a cero y, si es así, finaliza la cuenta (o si el segundo byte tiene el bit 7 a 1, el número es negativo). En la cuenta atrás es también necesario borrar la visualización del número anterior, antes de imprimir el nuevo, porque si se hace un cambio desde, por ejemplo, 1000 a 999, el "cero de las unidades" de 1000 podría permanecer en la pantalla, y el número que aparecería sería 99990.

El programa COUNTDEMO, disponible en cinta, muestra como utiliza la rutina de contador para contar el tiempo de reacción.

El programa BASIC es auto-explicativo y utiliza principalmente las instrucciones de impresión. Se utilizan sentencias DATA para guardar los parámetros de x,y,h,w y a$ para la impresión LARGE, según se trata en el Capítulo 2.



TIEMPO DE REACCION

CONTEO
61

S

EXCELENTE

OTRA VEZ? S/N

```
  12>PAPER 6:CLS
  15 DATA 87,143,1,2,"CONTAR"
  2Ø DATA 95,63,1,2,"LENTO"
  3Ø DATA 71,63,1,2"NORMAL"
  4Ø DATA 95,63,1,2"BUENO"
  5Ø DATA 63,63,1,2,"MUY BUENO"
  6Ø DATA 63,63,1,2,"EXCELENTE"
  8Ø DATA 47,63,1,2,"NINGUN INTENTO"
  9Ø DATA Ø,31,1,2"OTRA VEZ? s/n"
  95 DATA 55,125,5,2,"GRACIAS"
  96 DATA 1Ø3,79,3,2,"por"
  97 DATA 71,5Ø,5,2,"JUGAR"
 1ØØ DATA 24,79,5,2,"PARE LA CINTA"
 1Ø5 DATA 15,167,1,2,"TIEMPO DE REAC-
CION"
 11Ø DATA 15,164,2,2,"_____________
____"
 12Ø DATA 24,15,1,2,"PULSE CUALQUIER
TECLA"
 125 RESTORE 1ØØ
 13Ø FOR a=1 TO 4
 14Ø PAUSE 25
 16Ø INK 2: GO SUB 9ØØØ
 17Ø NEXT a
 18Ø PAUSE Ø: BORDER 5: PAPER 6:
INK Ø: CLS
 19Ø RESTORE 1Ø5
 22Ø INK Ø: GO SUB 9ØØØ
 25Ø PRINT AT 3,Ø;"Este programa demues-
tra el conteo con código/M combinando un
CONTADOR DE REACCION y una rutina de
conteo en código/M"
 26Ø PRINT '"La Spectrum seleccionará una
letra (A a Z) y la imprimirá PAPER 6: INK 6."
```

```
 27Ø PRINT '"El código/M cambiará INK a
INKØ, seleccionará CAPS LOCK y contará el
tiempo que tarda usted en pulsar la misma
tecla."
 28Ø PRINT '"Habrá un retardo aleatorio
antes de que aparezca la tecla seleccio-
nada."
 29Ø PRINT AT 2Ø,5;"PULSE CUALQUIER
TECLA PARA JUGAR"
 3ØØ IF INKEY$=<>"" THEN GO TO 3ØØ
 3Ø2 IF INKEY$="" THEN GO TO 3Ø2
 3Ø5 CLS
 31Ø RESTORE 1Ø5: INK 1: GO SUB 9ØØØ
 32Ø INK 5: GO SUB 9ØØØ
 33Ø RESTORE 15: INK 2: GO SUB 9ØØØ
 35Ø INK Ø: PLOT 1Ø2,114: DRAW 5Ø
,Ø:DRAW Ø-38:DRAW-5Ø,Ø: DRAW Ø,38
 38Ø LET z=INT (RND*26+65)
 39Ø LET x=111: LET y=111: LET h=
4: LET w=4: LET a$=CHR$ z
 4ØØ INK 6: LET c=USR 32393
 41Ø FOR a=1 TO RND*2ØØ+5Ø: NEXT a
 42Ø INK Ø
 43Ø LET c=USR 2376Ø
 44Ø LET a=PEEK 23728+256*PEEK 23
729
 45Ø IF a>=14Ø OR a=Ø THEN RESTO
RE 8Ø
 46Ø IF a<14Ø THEN RESTORE 2Ø
 47Ø IF a<12Ø THEN RESTORE 3Ø
 48Ø IF a<1ØØ THEN RESTORE 4Ø
 49Ø IF a<85 THEN RESTORE 5Ø
 5ØØ IF a<7Ø THEN RESTORE 6Ø
 51Ø GO SUB 9ØØØ
7ØØØ RESTORE 9Ø: INK 1: GO SUB 9ØØØ
7Ø1Ø IF INKEY$<>"" THEN GO TO 7Ø1Ø
7Ø2Ø IF INKEY$="" THEN GO TO 7Ø2Ø
7Ø3Ø LET d$=INKEY$
7Ø4Ø IF d$<>"y" THEN GO TO 8ØØØ
7Ø5Ø LET c=USR 23842: GO TO 35Ø
8ØØØ RESTORE 95
8ØØ5 POKE 23843,2Ø
8Ø1Ø LET c=USR 23842
8Ø15 POKE 23843,18
8Ø3Ø FOR a=1 TO 3
8Ø4Ø GO SUB 9ØØØ
8Ø5Ø NEXT a
8Ø6Ø STOP
9ØØØ READ x,y,h,w,a$
9Ø1Ø LET c=USR 32393
9Ø2Ø RETURN
98ØØ CLEAR 32334: LOAD " "CODE : G
O TO 5
99ØØ SAVE "REACTION" LINE 98ØØ
995Ø SAVE "LARGE"CODE 32335,265
  1Ø REM >z? \ STEP }~ CLEAR THE
N Ø OVER !?X~ RETURN 6 ?6Ø£: RETU
RN  ¿   NEXT ??? OR STEP +->? STEP
 ?                          BEEP >
 BEEP >?xLEN (? CLEAR THEN ?n ( O
UT  ! \PEEK    VERIFY  GO SUB CVAL
 \ CLEAR   THEN Øo<>
```

Org 2376Ø
Encontrar variable z

```
23760 3E 7A             ld a,122
Guardar en 23681
23762 Ø1 81 5C          ld bc,23681
23765 CD 7D 7E          call 32381

Establecer CAPS LOCK
23768 FD CB 3Ø DE set 3,(iy+48)

Intercambiar atributo de color

23772 21 ØØ 58          ld hl,22528
L1
23775 7E                ld a,(hl)
23776 FE 36             cp 54
PAPER 6 INK 6
23778 2Ø Ø2             jr nz,L2
2378Ø 36 3Ø             ld (hl),48
Reponer PAPER 6 INK Ø
L2
23782 23                inc hl
23783 7C                ld a,h
23784 FE 5B             cp 91
Fin de ATRIBUTOS        ?
23786 2Ø F3             jr nz,L1

Empezar la cuenta

23788 Ø1 ØØ ØØ          ld bc,Ø
Guardar cuenta en pila
L3
23791 C5                push bc
Poner cuenta en CALCULADORA
23792 CD 2B 2D          call 11563

Establecer PRINT AT 6,14
23795 3E Ø2             ld a,2
23797 CD Ø1 16          call 5633
238ØØ 3E 16             ld a,22


238Ø2 D7                rst 16
238Ø3 3E Ø6             ld a,6
238Ø5 D7                rst 16
238Ø6 3E ØE             ld a,14
238Ø8 D7                rst 16

Imprimir cuenta
238Ø9 CD E3 2D          call 11747

Obtener cuenta y sumarle 1
23812 C1                pop bc
23813 Ø3                inc bc

Examinar si cuenta < 65535
23814 78                ld a,b
23815 B1                or c
23816 28 ØF             jr z,END

Examinar tecla pulsada

23818 FD CB Ø1 6E bit 5, (iy+1)


23822 28 DF             jr z,L3
Encontrar INKEY$
23824 3A Ø8 5C          ld a,(2356Ø)
```

```
Comparar a$
23827 21 81 5C          ld hl,23681
23830 BE                cp (hl)
23831 20 D6             jr nz,L3
END
Guardar cuenta
23833 ED 43 B0 5C ld,(23728),bc
Recuperar minúsculas
23837 FD CB 30 9E res 3, (iy+48)
23841 C9                ret
Borrar 18 líneas inferiores
23842 06 12             ld b,18
23844 CD 44 0E          call 3652
23847 C9                ret
```

Programa 3.3

4 NUMEROS ALEATORIOS

Los números aleatorios son una parte esencial de la mayoría de los programas de juegos, tal como el retardo aleatorio y la elección aleatoria de letras del programa REACTION TIME (programa 3.3) del Capítulo 3. La asignación de un número aleatorio en código máquina es realmente muy fácil; puede abordarse de dos maneras, dependiendo del "azar/extensión" del número.

El primer método, que realmente no produce un número aleatorio verdadero, pero que es lo suficientemente bueno para muchos juegos, consiste en utilizar las variables FRAMES y SEED de forma distinta a como se usan normalmente. Es decir, si la extensión del número aleatorio es 0-1, 0-3, 0-7, 0-15, 0-31, 0-63, 0-127, 0-255 este método es válido. La rutina exige que el registro HL se cargue con el valor de SEED (23670); el registro DE se debe cargar con el byte alto de SEED/ y el byte bajo de FRAMES (23671); se suman los valores de los dos registros y se almacena la suma de nuevo en la variable SEED para el siguiente "número aleatorio". El registro A se carga después con el H o el L y se realiza un "AND a" para enmascarar los bits no deseados; de ahí la extensión de número aleatorio indicada anteriormente. Por ejemplo, el programa 4.1 nos da un ejemplo de INT (RND * 32) enmascarando los bits 7-5 del valor contenido en el registro A para conseguir números de 0 a 31.

```
org 2376Ø
1Ø números aleatorios
2376Ø Ø6 ØA          ld b,1Ø
L1
23762 C5             push bc
Load HL,(SEED)
23763 2A 76 5C       ld hl,(2367Ø)
23766 ED 5B 77 5C    ld de,(23671)
2377Ø 19             add hl,de

SEED modificado para siguiente número
23771 22 76 5C       ld (2367Ø),hl

Enmascarar BITS 7-5
23774 7D             ld a,1
23775 E6 1F          and 31

Guardar número aleatorio
23777 F5             push af

Set PRINT AT b,1Ø
23778 3E Ø2          ld a,2
2378Ø CD Ø1 16       call 5633
23783 3E 16          ld a,22
23785 D7             rst 16
23786 78             ld a,b
23787 D7             rst 16
23788 3E ØA          ld a,1Ø
2379Ø D7             rst 16

Obtener número aleatorio y ponerlo en
la PILA CALCULADORA
23791 F1             pop af
23792 CD 28 2D       call 1156Ø
23795 CD E3 2D       call 11747
Imprimirlo

Obtener cuenta de números
23798 C1             pop bc
```

```
repetir hasta 10 números
23799 10 D9            djnz L1
23801 C9               ret
```

Programa 4.1

```
PROGRAMA 1 EJEMPLO DE SALIDA IMPRESA

            6
            22
            23
            8
            9
            25
            24
            5
            1
            10
```

Como se dijo anteriormente esto es útil si se requiere un "número aleatorio" de esta extensión. Es un poco más complicado obtener INT (RND * 23), por ejemplo, mediante:

```
LD A, H
AND 15
LD H, A
LD A, L
AND 7
ADD A, H
```

Habrá observado que en el programa 4.1 el registro B se ha utilizado para guardar la cuenta del bucle (diez veces) y que este valor también se utiliza para la rutina PRINT AT b, 10;. El ejemplo impreso muestra lo aleatorio de la rutina.

Para producir un número aleatorio de la misma forma que con la orden INT (RND * n) del BASIC necesitaremos utilizar la rutina RND de la ROM, pero desgraciadamente esto no se puede hacer con una instrucción de llamada a RND ya que la rutina no vuelve después de ser ejecutada. La rutina ocupa las direcciones desde 9725 hasta 9765 de la ROM, y emplea la calculadora una vez más, para modificar el valor de SEED. Comienza extrayendo el valor existente de SEED, y termina con el nuevo valor duplicado en la parte superior de la pila. Se cambia el valor superior y se utiliza para actualizar SEED. Se modifica el valor que queda para dar un valor de 0 a 1 (pero no 1) que se utiliza para RND. Para utilizar esta rutina es necesario copiarla en RAM, y acceder a ella desde su programa como una subrutina. El programa 4.2 le muestra este método utilizado para obtener INT (RND * 12345) e imprimir 44 respuestas utilizando CHR$ CODE 6 para que la impresión se haga en dos columnas.

```
INT (RND*12345)

org 23760
CLS

23760 3E Ø2            ld a,2
23762 CD Ø1 16         call 5633
23765 CD 6B ØD         call 3435

44 números aleatorios

23768 Ø6 2C            ld b,44
L1
2377Ø C5               push bc

COPIA de ROM 9725 a 9765

23771 ED 4B 76 5C ld bc, (2367Ø)
23775 CD 2B 2D         call 11563

23778 EF               rst 4Ø
PILA (1)
defb 161

+
defb 15
PILA (75)
defb 52 55 22
*
defb 4
PILA (65537)
defb 52 128 65 Ø Ø 128
LITERALES
defb 5Ø 2
PILA (1)
defb 161
−
defb 3
Duplicar
defb 49
Fin RUTINA CALC
defb 56
Poner valor superior en BC
REGISTRO (INT Ø-65535)
23797 CD A2 2D         call 11682

Guardar nuevo SEED
238ØØ ED 43 76 5C ld (2367Ø),bc
Manipular BYTE EXPONENTE
238Ø4 7E               ld a,(hl)
238Ø5 A7               and a
238Ø6 28 Ø3            jr z,L2
238Ø8 D6 1Ø            sub 16
2381Ø 77               ld (hl),a
L2
PILA 12345
23811 Ø1 39 3Ø         ld bc,12345
23814 CD 2B 2D         call 11563
23817 EF               rst 4Ø
*
defb 4
LITERAL 'ENTERO'
defb 39
Fin RUTINA CALC
defb 56
Imprimir valor de PILA
23821 3E Ø2            ld a,2

23823 CD Ø1 16         call 5633
23826 CD E3 2D         call 11747
PRINT ,
23829 3E Ø6            ld a,6
23831 D7               rst 16

Obtener cuenta

23832 C1               pop bc
23833 1Ø BF            djnz L1
23835 C9               ret
```

Programa 4.2

EJEMPLO DE SALIDA IMPRESA DEL PROG. ANTERIOR

1857	3558
7666	7091
1037	3758
10296	6834
6415	12044
3058	7178
7727	11711
3057	7150
2940	10693

La rutina RND utiliza algunos literales nuevos que pueden servir también para otras rutinas:

- El LITERAL 161 es STACK NUMBER 1.

- El LITERAL 52 es STACK DATA (es normalmente un número expresado en forma reducida – ver Capítulo 7).
- El LITERAL 50 es n–mod–m.
- El LITERAL 2 es DELETE (el cociente).

Los dos últimos literales son de poca utilidad para programas de juegos. La rutina —CALL 11682— es utilizada por la ROM para recuperar el valor superior de la pila y colocar el valor entero en el registro BC. En este caso el valor entero se redondea por defecto o por exceso, según sea necesario, para obtener el número más próximo. Hay también una rutina similar —CALL 11733— que coloca en el registro A el valor superior de forma semejante. La rutina ROM copiada acaba en 23810.

La rutina desde 23811 a 23820 modifica el valor RND de la pila. Se apila el valor 12345 y después se multiplica por el valor RND. Después se utiliza un nuevo literal, el 39; éste obtiene cualquier valor decimal del resultado de redondear *por defecto* el valor de 12345 RND (como en el RND del BASIC), y así deja INT (RND * 12345) en la parte superior de la pila. Después se emplea la rutina PRINT VALUE para imprimir 44 valores en dos columnas, y los resultados muestran que estos valores son realmente aleatorios.

El programa de demostración 4.3 muestra como se utiliza realmente RND. En este caso se utiliza RND para imprimir rascacielos de altura al azar, con caracteres y colores aleatorios, y podría utilizarse en programas del tipo "bombas sobre la ciudad" (city bomb). Todos los UDGs para un programa completo han sido ya asignados y puede que desee desarrollar este programa. El programa "RNDDEMO" de la cinta utiliza el programa BASIC para conseguir la primera visualización de pantalla; esta se mantiene en la pantalla durante unos segundos, y después se utiliza la rutina en código máquina para imprimir la siguiente visualización, mostrando la velocidad del Código Máquina y también la función RND.

Un punto a considerar cuando se usa la instrucción RND del BASIC con llamadas USR en código máquina es que debe utilizar LET "variable" = USR "dirección" para llamar al código máquina, ya que RANDOMIZE USR "dirección" afectará al generador de números aleatorios. Puede utilizar también PRINT USR dirección, si no le importa que el valor contenido en el registro BC se imprima al volver al BASIC. En la rutina en código máquina, PAPER y los colores INK se establecen utilizando LD (IY + 83), 47 en vez de LD A,47: LD (IY + 83) A, como en ejemplos anteriores, ahorrando así memoria y mostrando la utilidad del registro IY como puntero. Esto se de-

muestra posteriormente con más detalle en el programa cuando IY se pone a 23296, comienzo de la memoria intermedia de la impresora, y es utilizado para LD (23296), 22 (AT) y LD (23299), 16 (INK). Sin embargo, cuando se altera el valor contenido en IY hay que tener cuidado de que si se hace una llamada ROM que emplee el registro IY, el valor debe reinicializarse a 23610 antes de hacer la llamada, o la rutina ROM se descontrolará y se puede originar una caída.

Antes de llamar a la rutina de número aleatorio se carga el registro A con el número RND * n y éste se almacena después en la dirección 23681 (una dirección no utilizada por las variables del sistema). Este número es después recogido en la rutina RND y utilizado para obtener INT (RND * n), el valor que se coloca en el registro A, llamando a 11733 antes de regresar de la rutina. La secuencia intermedia de la impresora es utilizada para guardar la información de PRINT en orden: AT (22), x, y, INK (16), b$, a$.

```
   1 REM >? PASO EL" BORRAR 65/>?
PASO?
  10 BORDER Ø: PAPER 5: INK 7: C L5
  20 FOR c=31 TO Ø STEP -1
  30 LET l =19-INT (RND*1Ø)
  40 LET a$=CHR$ INT (RND*2+148)
  45 LET b$=CHR$ INT (RND*4)
  50 PRINT AT L -1,c;"▲"
  60 FOR a=l TO 2Ø
  70 PRINT AT a,c;CHR$ 16+b$+a$
  80 NEXT a
  90 PRINT AT a, c;"■"
 100 NEXT c
 110 LET x=Ø: LET y=Ø: LET a$="
"
 13Ø PRINT AT x,y;a$
 14Ø PRINT #Ø; "■ COMBUSTIBLE 1ØØØØ ■
BOMBAS
1ØØ ■ DISPAROS 1ØØ ■"
 15Ø RETURN
 5ØØ DATA 24,79,5,2,"PARE LA CINTA"
 51Ø DATA 15, 167, 1, 2, "NUMEROS ALEATO-
RIOS"
 52Ø DATA 15,164,1,2,"___________________"
 525 DATA 24,15,1,2,"PULSE CUALQUIER TE-
CLA"
 53Ø DATA 47,164,2,4,"BASIC"
 54Ø DATA 31,164,3,2,"CODIGO MAQUINA"
 6ØØ BORDER 4: PAPER 6: INK Ø: C L5
 6Ø5 RESTORE 5ØØ
 61Ø FOR a=1 TO 4
 62Ø INK a: GO SUB 8ØØØ
 63Ø PAUSE 5Ø: NEXT a
 64Ø PAUSE Ø: CLS
 645 PRINT INK Ø;AT 8,Ø; "Este programa
muestra la rutina RND en código Máquina y
compara la velocidad de las versiones BASIC y
código Máquina.
 65Ø RESTORE 525
```

```
 655 PAUSE 100
 660 FLASH 1: INK 2: PAPER 7: GO
 SUB 8000: FLASH 0
 670 PAUSE 0: CLS
1000 LET z=1
1010 GO SUB 2000
1020 GO SUB 10
1025 LET z=z+1
1030 GO SUB 2000
1040 GO SUB 3000
1045 LET z=z+1
1050 GO TO 1010
2000 FOR a=1 TO 400: NEXT a: CLS

2010 PAPER 8: INK 0: RESTORE
530+(10*(z/2=INT (z/2))): GO SUB 8000
2020 FOR a=1 TO 200: NEXT a: RETURN
3000 LET c=USR 23760: GO TO 110
8000 READ x,y,h,w,a$
8010 LET c=USR 32393
8020 RETURN
9998 CLEAR 32334: LOAD "Udg"CODE USR
"a": LOAD ""CODE : GO TO 500
9999 SAVE "RNDDEMO" LINE 9998: S
AVE "Udg"CODE USR "a",168: SAVE
"LARGE"CODE 32335,265
```

```
org 23760

BORDER 0

23760 3E 00          ld a,0
23762 CD 9B 22       call 8859

PAPER 5 INK 7

23765 FD 36 53 2F ld (iy+83),47

CLS

23769 3E 02          ld a,2
23771 CD 01 16       call 5633
23774 CD 6B 0D       call 3435

Establecer IY como PUNTERO

23777 FD 21 00 5B ld iy,23296

23781 FD 36 00 16 ld (iy+0),22
23785 FD 36 03 10 ld (iy+3),16

Reinicializar IY
23789 FD 21 3A 5C ld iy,23610

Contador de columnas

23793 3E 1F          ld a,31
L1
23795 32 02 5B       ld (23298),a

19-INT (RND*10)

23798 3E 0A          ld a,10
23800 32 81 5C       ld (23681),a
```

```
23803 CD 61 5D          call RND
23806 6F                ld l,a
23807 3E 13             ld a,19
23809 95                sub 1
23810 32 01 5B          ld (23297),a
```

INT (RND*2+148)

```
23813 3E 02             ld a,2
23815 32 81 5C          ld (23681),a
23818 CD 61 5D          call RND
23821 C6 94             add a,148
23823 32 05 5B          ld (23301),a
```

INT (RND*4)

```
23826 3E 04             ld a,4
23828 32 81 5C          ld (23681),a
23831 CD 61 5D          call RND
23834 32 04 5B          ld (23300),a
```

PRINT AT x−1,y CHR$ 147

```
23837 3E 02             ld a,2
23839 CD 01 16          call 5633
23842 3E 16             ld a,22
23844 D7                rst 16
23845 3A 01 5B          ld a,(23297)
23848 3D                dec a
23849 D7                rst 16
23850 3A 02 5B          ld a,(23298)
23853 D7                rst 16
23854 3E 93             ld a,147
23856 D7                rst 16
L2
```

```
FOR a=x TO 20
PRINT AT a,y CHR$       16+b$+a$
23857 11 00 5B          ld de,23296
23860 01 06 00          ld bc,6
23863 CD 3C 20          call 8252
23866 3A 01 5B          ld a, (23297)
23869 3C                inc a
23870 32 01 5B          ld (23297),a
23870 32 01 5B          ld (23297),a
23873 FE 15             cp 21
NEXT a
23875 20 EC             jr nz,L2
```

PRINT AT a,y CHR$ 153

```
23877 3E 99             ld a,153
23879 32 05 5B          ld (23301),a
23882 3E 07             ld a,7
23884 32 04 5B          ld (23300),a
23887 11 00 5B          ld de,23296
23890 01 06 00          ld bc,6
23893 CD 3C 20          call 8252
23896 3A 02 5B          ld a,(23298)
23899 3D                dec a
23900 FE FF             cp 255
```

SIGUIENTE COLUMNA

```
23902 20 93           jr nz,L1
23904 C9              ret
```

RUTINA RND DE ROM

```
RND
23905 ED 4B 76 5C ld bc, (23670)
23909 CD 2B 2D        call 11563
23912 EF              rst 40
defb 161 15 52 55 22 4 52 128
defb 65 0 0 128 50 2 161 3 49 56
23931 CD A2 2D        call 11682
23934 ED 43 76 5C     ld (23670),bc
23938 7E              ld, a, Chl)
23939 A7              and a
23940 28 03           jr z,L3
23942 D6 10           sub 16
23944 77              ld (hl),a
```

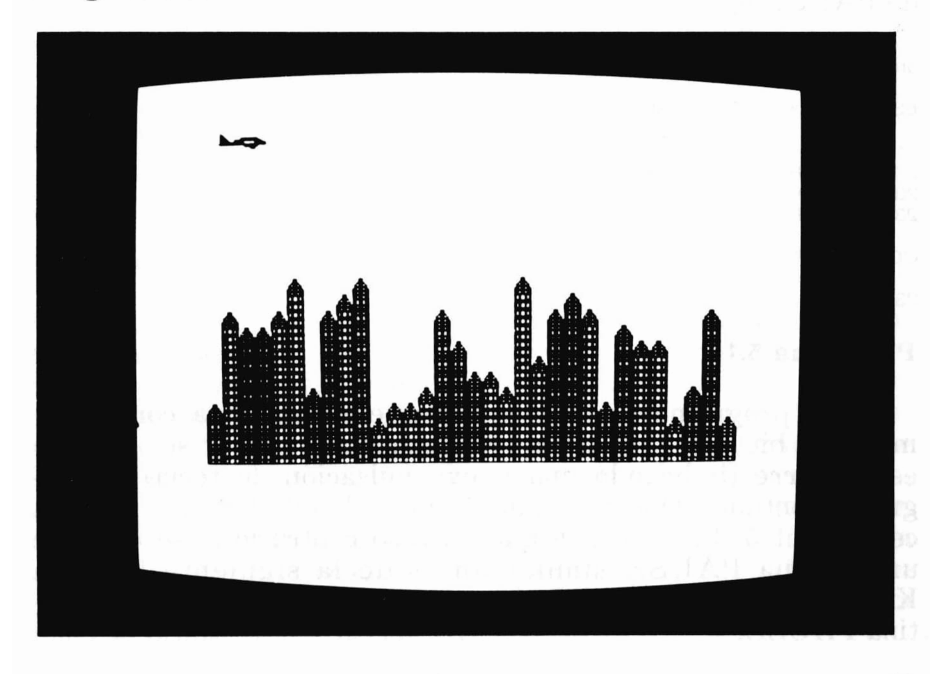
INT (RND*PEEK (23681))

```
L3
23945 3A 81 5C        ld a,(23681)
23948 CD 28 2D        call 11560
23951 EF              rst 40
defb 4 39 56
```

VALOR en A

```
23955 CD D5 2D        call 11733
23958 C9              ret
```

Programa 4.3



5 EL TECLADO

Hay dos métodos para leer el teclado en código máquina. Ambos son útiles, pero el método elegido depende del tipo de programa.

Utilización de la variable LAST KEY

La ROM de la Spectrum contiene una rutina KEYSCAN y DECODE a la cual se accede 50 veces durante cada segundo. La rutina también actualiza el contador FRAMES.

El código de carácter de la última tecla pulsada es almacenado en la variable LAST KEY (23560) y si es una nueva tecla, entonces el bit 5, (IY + 1) se pone a uno. Incidentalmente la orden HALT espera a que se haga la siguiente llamada a KEYSCAN antes de continuar con la rutina en código máquina.

Con la anterior información podemos escribir el equivalente de PAUSE Ø.

```
org 2376Ø

ESPERAR A QUE SE PULSE TECLA

L1
2376Ø FD CB Ø1 6E bit 5,(iy+1)
23764 28 FA             jr z,L1
23766 FD CB Ø1 AE res 5(iy+1)

CONTINUAR CON PROGRAMA

2377Ø C9                ret
```

Programa 5.1

En el programa 5.1 puede ver que se examina constantemente el bit 5, (IY + 1) para ver cuando es uno, y sólo cuando esto ocurre (indicando una nueva pulsación de tecla) el programa continúa. Observar que después de ello hay que poner a cero el bit 5 (IY + 1), ya que en caso contrario si se utilizase una rutina PAUSE similar antes de la siguiente llamada KEYSCAN, el bit 5 podría estar a uno y pasarse por alto la rutina PAUSE.

Utilización de IN A, (C)

Es idéntica a la orden IN del BASIC leyendo la mitad de la fila especificada por el valor que contiene el registro BC, y se trata en el manual de la Spectrum. El valor que queda en A indica qué teclas se pulsan de esa mitad de fila. Cada mitad de fila contiene cinco teclas, y los bits 4-0 de A contienen el estado de esas teclas (los bits 7-5 están normalmente a uno), el bit 0 se utiliza para indicar el estado de la "tecla externa" en la mitad de fila, y el bit 4 el estado de la "tecla interna".

Cuando no se pulsa ninguna tecla todos los bits están a uno, y A contendrá 255. Si se pulsa una tecla su bit se pondrá a cero. Por ejemplo, si se pulsase la tecla Ø entonces el bit 0 se pondría a cero y A podría valer 254. Puede ver que con este método tenemos la posibilidad de leer más de una tecla a la vez, cuando se pulsan varias al mismo tiempo, examinando el valor de A o los bits individuales de A. Esto podría ser útil en un juego para dos o más jugadores o para mover cosas en dos direcciones (por ejemplo hacia arriba y hacia abajo) al mismo tiempo. El programa 5.2 muestra la orden IN utilizada para esperar hasta que se pulse la tecla Ø.

```
org 2376Ø

LEER MITAD FILA 6-Ø

L1
2376Ø 1Ø FE EF        ld, bc,61438
23763 ED 78           in a,(c)

EXAMINAR SI TECLA Ø ES PULSADA

23765 CB47            bit Ø,a
23767 C8              ret z
23768 18 F6           jr L1
```

Programa 5.2

La llamada a KEYSCAN de la ROM se hace 50 veces durante cada segundo incluso *durante nuestro propio programa en código máquina* y, por ello, el programa es más lento. Si necesitamos una velocidad extra, por ejemplo, para la visualización enrollada de pixel, podemos parar esta interrupción de nuestro programa utilizando la instrucción DI. Sin embargo, esto significa que mientras esté deshabilitada la interrupción no podemos utilizar la rutina LAST KEY para leer el teclado (a menos que hagamos nosotros la llamada a la rutina ROM) y debemos recurrir a la instrucción IN. El programa 5.3 muestra como se utiliza la instrucción DI.

```
org 23760

DESHABILITAR

23760 F3                di

LEER MITAD FILA 6-0

L1
23761 01 FE EF          ld bc,61438
23764 ED 78             in a,(c)

EXAMINAR SI TECLA 0 ES PULSADA

23766 CB 47             bit 0,a
23768 20 F7             jr nz,L1

HABILITAR

23770 FB                ei

23771C9                 ret
```

Programa 5.3

Observe que la instrucción EI debe utilizarse antes de volver al BASIC, o bien quedará inutilizado el teclado. Intente probar el programa sin la instrucción EI y vea qué sucede. No se producirá ningún daño, pero la Spectrum tendrá que desenchufarse y volverse a conectar para recuperar la función del teclado.

El programa 5.4 muestra la forma de obtener PAUSE n para conseguir un retardo de tiempo 0, como en el BASIC; continúa si se pulsa una tecla. El programa utiliza el hecho de que el examen de tecla ocurre cada 0,02 segundos y de que HALT espera a que se examine el teclado.

```
org 23760
Establecer RETARDO 10 SEG

23760 01 F4 01          ld bc,500
L1
LAZO DE PAUSA
23763 76                halt

EXAMINAR SI SE PULSA TECLA

23764 FD CB 01 6E bit 5,(iy+1)
23768 20 05             jr nz,L2
DISMINUIR CONTADOR
23770 0B                dec bc
23771 78                ld a,b
23772 B1                or c
23773 20 F4             jr nz, L1
L2
23775 FD CB 01 AE       res 5,(iy+1)

23779 C9                ret
```

Programa 5.4

El programa 5.5 se encuentra en la cinta "KEYDEMO" y hace uso del método LAST KEY de lectura del teclado para producir un programa de mecanografía simple que permite el borrado y la modificación de lo escrito.

La posición de impresión actual es indicada por un cursor parpadeante. Observe que CAPS SHIFT y "2" mueve la posición de impresión a la siguiente PRINT COMMA TAB; CAPS SHIFT y "5" produce un espacio hacia atrás (pero observe el error de ROM por el cual no se puede volver un espacio desde el comienzo de la línea 1 hasta el final de línea 0), y ENTER produce un salto de línea. La tecla BREAK es inoperante en código máquina por lo que he utilizado "STOP" para volver al BASIC.

INSTRUCCIONES

Este programa demuestra el método de lectura del teclado en código/M utilizando la variable LAST KEY y examinando el BIT 5 de FLAGS para ver si es una NUEVA tecla.

El cursor puede volver atrás usando la tecla 5 de cursor para corregir o borrar cuando se escribe.

ENTER le dará una nueva línea, y como la tecla BREAK le da un SPACE, utilice 'STOP' para volver al BASIC.

El código /M usa 43 bytes.

PULSE CUALQUIER TECLA

Como ocurre con la instrucción INKEY$ del BASIC no se leen ni las teclas de doble-cambio ni las teclas de modo gráfico. Si desea examinar o imprimir teclas con doble-cambio, debe establecer una tabla DATA de caracteres sin cambio y de sus equivalentes con doble-cambio. Entonces si LDA (LAST KEY) regresa con el valor 14 en A, indicando que se han pulsado dos teclas con cambio, se necesitará una rutina para leer la siguiente tecla pulsada y convertir su carácter sin cambio en su correspondiente con cambio utilizando la tabla DATA.

```
10 REM >? STEP ?
20>DATA 24,79,5,2, "PARE LA CINTA"
30 DATA 7,170,2,3,"MAQUINA DE ES-
CRIBIR"
40 DATA 7,150,2,3, "__________"
50 DATA 24, 15, 1, 2"PULSE CUAL-
QUIER TECLA"

60 DATA 39,170,1,2,"INSTRUCCIONES"
65 RESTORE
70 BORDER 4: PAPER 6: CLS
80 FOR A=1 TO 4
85 PAUSE 20
90 INK A: PAPER 8: GO SUB 8000
100 NEXT A
110 PAUSE 0: PAPER 7: CLS
120 INK 2: PAPER 8: 60 SUB 8000
130 INK 0: PRINT AT 3,0; "Este programa
demuestra el método de lectura del teclado
en código/M utilizando la variable LAST KEY
```

y examinando el BIT'5 de FLAGS para ver si es una NUEVA tecla.

```
140 PRINT '"El cursor puede volver atrás usando la tecla 5 del cursor para corregir o borrar cuando se escribe."
150 PRINT '"ENTER le dará una nueva línea y como la tecla BREAK le da un SPACE, use 'STOP' para volver al BASIC."
160 PRINT ' "El código /M usa 43 bytes"
200 RESTORE 50: INK 2: GO SUB 8000
210 PAUSE 0: BRIGHT 1: CLS
220 INK 0: RANDOMIZE USR 23760
230 STOP
8000 READ x,y,h,w,a$
8010 RANDOMIZE USR 32393
8020 RETURN
9990 CLEAR 32334: LOAD ""CODE : GO TO 20
9999 SAVE "LARGE" CODE 32335,265
```

```
org 23760

Abrir canal 2

23760 3E 02          ld a,2
23762 CD 01 16       call 5633

ESTABLECER OVER 1

L1
23765 FD 36 57 03 ld (iy+87),3

CURSOR PARPADEA EN LAZO DE ESPERA

23769 3E 8F          ld a,143
23771 D7             rst 16
23772 3E 08          ld a,8
23774 D7             rst 16
23775 3E 8F          ld a,143
23777 D7             rst 16

23778 3E 08          ld a,8
23780 D7             rst 16

ESPERAR PULSACION DE TECLA

23781 FD CB 01 6E bit 5, (iy+1)
23785 28 EA          jr z,L1
23787 FD CB 01 AE    res 5,(iy+1)

ESTABLECER OVER 0

23791 FD 36 57 00 ld (iy+87),0

OBTENER LAST KEY (ULTIMA TECLA)

23795 3A 08 5C       ld a,(23560)

EXAMINAR SI ES 'STOP'

23798 FE E2          cp 226
23800 C8             ret z

IMPRIMIR CHR$

23801 D7             rst 16
23802 18 D9          jr L1
```

Programa 5.5

6 MOVIMIENTO

Ahora que somos capaces de escribir en la pantalla y de leer el teclado podemos buscar métodos de animación. Esto es, por supuesto, parte esencial de cualquier programa de juegos en "tiempo real". En este Capítulo estudiaremos métodos para dar la impresión de movimiento:

1. Movimiento de primer plano —barcos, marcianos, gente, etc.
2. Movimiento del fondo —estrellas, planetas, fragmentos de cielo, etc.
3. Movimiento de ambos, 1 y 2.

El tipo de movimiento empleado por la mayoría de las personas que hacen software es de "cuadro de carácter"; éste permite que los programas sean rápidos y la utilización de todos los colores. La pantalla se divide en 24 filas y 32 columnas, y, cuando se mueve un carácter, cambia desde su posición actual hasta la fila/columna adyacente; este movimiento es tosco pero no a sacudidas. Si se emplea el movimiento de pixel entonces, especialmente cuando se trata de movimientos de primer término, los caracteres deben redibujarse. Si ha utilizado la orden PLOT del BASIC se habrá dado cuenta de que normalmente nos quedamos restringidos al uso de dos colores (PAPER e INK).

Movimiento de primer plano de un cuadro de carácter

El primer paso es definir (o inicializar) la posición de comienzo del carácter y, utilizando cualquiera de los métodos citados en el Capítulo 5, leer el teclado, y actualizar la posición del carácter de acuerdo con ello. La actualización de la posición del carácter es realizada en un "grupo duplicado" de datos de posiciones de comienzo. Cuando la actualización ha sido completada habrá dos grupos de datos: el primero contendrá la posición de pantalla actual y el segundo la nueva posición. Se llevan a cabo algunas pruebas, por ejemplo, para ver si la nueva posición está aún en la pantalla, y se realiza la corrección. Para conseguir la

impresión de movimiento de parpadeo-libre, sacamos el carácter existente y lo volvemos a visualizar *inmediatamente* en su nueva posición. El método que yo uso generalmente consiste en utilizar OVER 1 y REPRINT para volver a visualizar el carácter en la posición *anterior;* de esta forma, sacándolo, y utilizando todavia OVER 1 volvemos a repintar en la *nueva* posición. Esto nos permitirá mover el carácter sobre el fondo sin borrarlo.

```
org 2376Ø
DESHABILITAR TECLADO

2376Ø F3                di
23761 18 18             jr START
DATOS UDG

DATA 1
CHR$ 144
defb Ø Ø 15 24 49 226 225 224
CHR$ 145
defb 24 6Ø 255 255 153 255 231
defb 6Ø
CHR$ 146
defb Ø Ø 24Ø 24 14Ø 71 135 7
EMPEZAR

ESTABLECER UDG's

23787 ED 5B 7B 5C       ld de,(23675)

23791 21 D3 5C          ld hl,DATA1
23794 Ø1 18 ØØ          ld bc,24
23797 ED BØ             ldir
23799 18 ØA             jr BEGIN

Print CHR$ DATA
AT x,y INK 8 PAPER 8
CHR$ 144 145 146

DATA2
defb 22 Ø Ø 16 8 17 8
defb 144 145 146
BEGIN

BORDER 5
23811 FD 36 ØE 28 ld (iy+14),4Ø

PAPER 1 INK 7
23815 FD 36 53 ØF ld (iy+83),15
CLS
23819 3E Ø2             ld a,2


23821 CD Ø1 16          call 5633
23824 CD 6B ØD          call 3435

IMPRIMIR SERIE

23827 3E Ø2             ld a,2
23829 CD Ø1 16          call 5633
23832 11 F9 5C          ld de,DATA2
23835 Ø1 ØA ØØ          ld bc,1Ø
23838 CD 3C 2Ø          call 8252
L7

TRANSFERIR DATA 2 A STORE

23841 11 ØØ 5B          ld de,23296
23844 21 F9 5C          ld hl,DATA2
23847 Ø1 ØA ØØ          ld bc,1Ø
2385Ø ED BO

ldir

OBTENER x,y DE STORE

23852 2A Ø1 5B          ld hl,(23297)

EXAMINAR SI SE PULSA TECLA

23855 Ø1 FE EF          ld bc,61438
23858 ED 78             in a,(c)
2386Ø CB 47             bit Ø,a
23862 2Ø Ø1             jr nz,L1
23864 24                inc h
L1
23865 CB 4F             bit 1,a
23867 2Ø Ø1             jr nz,L2
23869 25                dec h
L2
2387Ø Ø1 FE F7          ld bc,63486
23873 ED 78             in a,(c)
23875 CB 47             bit Ø,a
23877 2Ø Ø1             jr nz,L3
23879 2D                dec l
L3
2388Ø Ø1 FE FB          ld bc,6451Ø

23883 ED 78             in a,(c)
23885 CB 47             bit Ø,a
23887 2Ø Ø1             jr nz,L4
23889 2C                inc 1
L4

EXAMINAR SI ESTA QUIETO EN PANTALLA

2389Ø 7D                ld a,l
23891 FE 16             cp 22
23893 2Ø Ø3             jr nz,L5
23895 2D                dec 1
23896 18 Ø5             jr L6
L5
```

```
23898 FE FF          cp 255
23900 20 01          jr nz,L6
23902 2C             inc l
L6
23903 7C             ld a,h
23904 FE 1E          cp 30
23906 20 03          jr nz,L8

23908 25             dec h
23909 18 05          jr L9
L8
23911 FE FF          cp 255
23913 20 01          jr nz,L9
23915 24             inc h
L9

EXAMINAR SI NO SE MUEVE

23916 ED 5B 01 5B    ld de,(23297)
23920 AF             xor a
23921 E5             push hl
23922 ED 52          sbc hl,de
23924 E1             pap hl
SI NO SE MUEVE ENTONCES PASAR
POR ALTO PRINK OVER 1
23925 28 2C          jr z,L12

GUARDAR NUEVOS PARAMETROS x,y

23927 22 01 5B       ld (23297),hl

ESTABLECER OVER 1

23930 FD 36 57 03 ld (iy+87),3

BORRAR POSICION ANTIGUA

23934 11 F9 5C       ld de,DATA2
23937 01 0A 00       ld bc,10
23940 CD 3C 20 call 8252

REIMPRIMIR EN NUEVA POSICION

23943 11 00 5B       ld de,23296
23946 01 0A 00       ld bc,10
23949 CD 3C 20       call 8252

ACTUALIZAR DATA2 PARA SIGUIENTE VEZ

23952 11 F9 5C       ld de,DATA2

23955 21 00 5B       ld hl,23296
23958 01 0A 00       ld bc,10
23961 ED B0          ldir

LAZO DE RETARDO

23963 21 88 13       ld hl,5000
L11
23966 2B             dec hl
23967 7C             ld a,h
23968 B5             or 1
23969 20 FB          jr nz,L11

EXAMINAR SI PULSA 'SPACE'

L12
23971 01 FE 7F       ld bc,32766
23974 ED 78          in a,(c)
23976 CB 47          bit 0,a
23978 C2 21 5D       jp nz,L7

REINICIALIZAR OVER 0

23981 FD 36 57 00    ld (iy+87),0

ACTIVAR TECLADO

23985 FB             ei
23986 C9             ret
```

Programa 6.1A

El programa 6.1A muestra el método explicado anteriormente. Se mueve un carácter, parecido a una nave espacial, sobre la pantalla, utilizando las teclas 9/0 para izquierda/derecha y 1/Q para arriba/abajo. Con la "barra de espaciado" volvemos al BASIC. La nave espacial puede moverse en dos direcciones a la vez y puede volar por la pantalla sin cambiar el fondo que anteriormente usted haya establecido. El carácter se imprime con INK 8; PAPER 8; y así tomará los colores de los cuadros según se defina el atributo de cada cuadro. La velocidad de movimiento es controlada con el lazo de retardo y se hace una prueba para ver si la nave está estacionaria. Si lo está, entonces se omite la rutina de revisualización ya que podría originar el parpadeo de la nave por el borrado y la revisualización continúa en la misma posición. Puede modificar el programa

para hacer posible que la nave vuele fuera de la pantalla y reaparezca por el otro lado, poniendo a cero "y" cuando la nave sobrepase los parámetros de la pantalla.

El movimiento de carácter múltiple es básicamente lo mismo que el anterior y cada carácter tiene su propio grupo de PRINT AT DATA. La rutina en código máquina es muy rápida, como puede demostrarse utilizando LD HL, 1 para el lazo de retardo, porque la TV no puede imprimir todas las posiciones cuando el carácter se mueve. Encontrará que la mayoría de los programas necesitan algún tipo de lazo de retardo para mantener la imagen durante un momentos antes de que cambie.

Movimiento de fondo de un cuadro de carácter

Siempre que el movimiento de primer plano requiera guardar en memoria la posición del carácter y que siga la pista de los movimientos, el movimiento de fondo requerirá normalmente el desplazamiento de la visualización completa de la pantalla (o bloques de ella) en una dirección conocida sin tener que saber exactamente donde está cada cosa.

Este movimiento se realiza usualmente en una de las cuatro direcciones —arriba, abajo, izquierda y derecha— y requiere normalmente ambas cosas, los caracteres y sus atributos. Si un fondo de pantalla es desplazado por una posición de carácter tendremos una posterior elección concerniente a los caracteres desplazados fuera de la pantalla. Podemos eliminarlos e imprimir espacios en la fila/columna formada al otro lado de la pantalla, o podemos almacenarlos con sus atributos y reimprimirlos en su lugar correspondiente al otro lado de la pantalla, produciéndose así un efecto de enrosque, que es útil para los programas de juegos.

Los siguientes programas del 6.1B al 6.16 muestran un ROLL y un SCROLL en las cuatro direcciones; en primer lugar los atributos y después los caracteres de pantalla. Estos programas se pueden modificar fácilmente para efectuar un ROLL y un SCROLL para posiciones de pantalla, quedando el resto inmóvil. Observe la capa peculiar de la visualización de la pantalla que se divide en tres bloques de ocho filas. Cada fila se divide a su vez en ocho líneas de 32 bytes por línea. El mapa de memoria empieza en la fila 1 línea 1 y continúa con los 32 bytes de esa línea; después cambia a la fila 2 línea 1, fila 3 línea 1 y así sucesivamente hasta la fila 8 línea 1 completando el mapa de 256 bytes. Entonces se vuelve a saltar a la fila 1 línea 2, fila 2

línea 2, fila 3 línea 2, etc., y se termina con la fila 8 línea 8 llenando el mapa de 2K bytes.

Después cambiamos a las ocho filas del medio que son tratadas de la misma forma, y finalmente a las ocho filas últimas que también se direccionan así. Así se produce un "roll" y "scroll" de carácter un poco complicado y requiere el movimiento de 6K bytes.

```
org 2376Ø
2376Ø 11 ØØ 57       ld de,22528
23763 21 2Ø 58       ld hl,2256Ø
23766 Ø1 EØ Ø2       ld bc,736
23769 ED BØ          ldir
23771 Ø6 2Ø          ld b,32
23773 3A 8D 5C       ld a,(23693)
L2
23776 12             ld (de),a
23777 13             inc de
23778 1Ø FC          djnz L2
2378Ø C9             ret
```

Programa 6.1B

```
org 2376Ø
2376Ø 11 FF 5A       ld de,23295
23763 21 DF 5A       ld hl,23263
23766 Ø1 EØ Ø2       ld bc,736
23769 ED B8          lddr
23771 Ø6 2Ø          ld b,32
23773 3A 8D 5C       ld a,(23693)
L7
23776 12             ld (de),a
23777 13             inc de
23778 1Ø FC          djnz L7
2378Ø C9             ret
```

Programa 6.2

```
org 2376Ø
2376Ø Ø6 18          ld b,24
23762 21 ØØ 58       ld hl,22528
L12
23765 E5             push hl
23766 D1             pop de
23767 C5             push bc
23768 23             inc hl
23769 Ø1 1F ØØ       ld bc,31
23772 ED BØ          ldir
23774 3A 8D 5C       ld a,(23693)
23777 12             ld (de),a
23778 C1             pop bc
23779 1Ø FØ          djnz L12
23781 C9             ret
```

Programa 6.3

```
org 2376Ø
2376Ø Ø6 18          ld b,24
23762 21 FF 5A       ld hl,23295
L14
23765 E5             push hl
23766 D1             pop de
23767 C5             push bc
23768 2B             dec hl
23769 Ø1 1F ØØ       ld bc,31
23772 ED B8          lddr
23774 3A 8D 5C       ld a,(23693)
23777 12             ld (de),a
23778 C1             pop bc
23779 1Ø FØ          djnz L14
23781 C9             ret
```

Programa 6.4

```
org 2376Ø
2376Ø 21 ØØ 58       ld hl,22528
23763 E5             push hl
23764 D1             pop de
23765 Ø6 2Ø          ld b,32
L16
23767 7E             ld a,(hl)
23768 F5             push af
23769 23             inc hl
2377Ø 1Ø FB          djnz L16
23772 Ø1 EØ Ø2       ld bc,736
23775 ED BØ          ldir
23777 Ø6 2Ø          ld b,32
L17
23779 F1             pop af
2378Ø 2B             dec hl
23781 77             ld (hl),a
23782 1Ø FB          djnz L17
23784 C9             ret
```

Programa 6.5

```
org 2376Ø
2376Ø 21 FF 5A      ld hl,23295
23763 E5            push hl
23764 D1            pop de
23765 Ø6 2Ø         ld b,32
L24
23767 7E            ld a,(hl)
23768 F5            push af
23769 2B            dec hl
2377Ø 1Ø FB         djnz L24
23772 Ø1 EØ Ø2      ld bc,736
23775 ED B8         lddr
23777 Ø6 2Ø         ld b,32
L25
23779 F1            pop af
2378Ø 23            inc hl
23781 77            ld (hl),a
23782 1Ø FB         djnz L25
23784 C9            ret
```

Programa 6.6

```
org 2376Ø
2376Ø Ø6 18         ld b,24
23762 21 ØØ 58      lh hl,22528
L32
23765 E5            push hl
23766 D1            pop de
23767 C5            push bc
23768 7E            ld a,(hl)
23769 23            inc hl
2377Ø Ø1 1F ØØ      ld bc,31
23773 ED BØ         ldir
23775 12            ld (de),a
23776 C1            pop bc
23777 1Ø F2         djnz L32
23779 C9            ret
```

Programa 6.7

```
org 2376Ø
2376Ø Ø6 18         ld b,24
23762 21 FF 5A      ld hl,23295
L33
23765 E5            push hl
23766 D1            pop de
23767 C5            push bc
23768 7E            ld a,(hl)
23769 2B            dec hl
2377Ø Ø1 1F ØØ      ld bc,31
23773 ED B8         lddr
23775 12            ld (de), a
23776 C1            pop bc
23777 1Ø F2         djnz L33
23779 C9            ret
```

Programa 6.8

```
or 2376Ø
2376Ø Ø6 CØ         ld b,192
23762 21 ØØ 4Ø      ld hl,16384
L1
23765 E5            push hl
23766 D1            pop de
23767 C5            push bc
23768 7E            ld a,(hl)
23769 23            inc hl
2377Ø Ø1 1F ØØ      ld bc,31
23773 ED BØ         ldir
23775 12            ld (de), a
23776 C1            pop bc
23777 1Ø F2         djnz L1
23779 C9            ret
```

Programa 6.9

```
org 2376Ø
2376Ø Ø6 CØ         ld b,192
23762 21 FF 57      ld hl,22527
L1
23765 E5            push hl
23766 D1            pop de
23767 C5            push bc
23768 7E            ld a,(hl)
23769 2B            dec hl
2377Ø Ø1 1F ØØ      ld bc,31
23773 ED B8         lddr
23775 12            ld (de), a
23776 C1            pop bc
23777 1Ø F2         djnz L1
23779 C9            ret
```

Programa 6.10

```
org 2376Ø
2376Ø 21 ØØ 4Ø          ld hl,16384
23763 AF                xor a
23764 Ø6 CØ             ld, b,192
L13
23766 C5                push bc
23767 E5                push hl
23768 D1                pop de
23769 23                inc hl
2377Ø Ø1 1F ØØ          ld, bc,31
23773 ED BØ             ldir
23775 12                ld (de), a
23776 C1                pop bc
23777 1Ø F3             djnz L13
23779 C9                ret
```

Programa 6.11

```
org 2376Ø
L2
2376Ø 21 ØØ 4Ø          ld hl,16384
23763 11 2Ø ØØ          ld de,32
23766 Ø6 CØ             ld b,192
L1
23768 36 ØØ             ld (hl),Ø
2377Ø 19                add hl, de
23771 1Ø FB             djnz L1
23773 21 Ø1 4Ø          ld hl,16385
23776 11 ØØ 4Ø          ld de,16384
23779 Ø1 FF 17          ld bc,6143
23782 ED 8Ø             ldir
23784 EB                ex de, hl
23785 36 ØØ             ld (hl),Ø

23787 C9                ret
```

Programa 6.11A

```
org 2376Ø
2376Ø 21 FF 57          ld hl,22527
23763 AF                xor a
23764 Ø6 CØ             ld b,192
L14
23766 C5                push bc
23767 E5                push hl
23768 D1                pop de
23769 2B                dec hl
2377Ø Ø1 1F ØØ          ld bc,31
23773 ED B8             lddr
23775 12                ld (de), a
23776 C1                pop bc
23777 1Ø F3             djnz L14
23779 C9                ret
```

Programa 6.12

```
org 2376Ø
2376Ø 11 ØØ 4Ø          ld de,16384
23763 21 2Ø 4Ø          ld hl,16416
23766 Ø1 EØ Ø7          ld bc,2Ø16
23769 ED BØ             ldir
23771 Ø6 Ø8             ld b,8
23773 21 ØØ 48          ld hl,18432
23776 11 EØ 4Ø          ld de,166Ø8
L3
23779 C5                push bc
2378Ø E5                push hl
23781 D5                push de
23782 Ø1 2Ø ØØ          ld bc,32
23785 ED BØ             ldir
23787 D1                pop de
23788 E1                pop hl
23789 C1                pop bc
2379Ø 14                inc d
23791 24                inc h
23792 1Ø F1             djnz L3

23794 11 ØØ 48          ld de,18432
23797 21 2Ø 48          ld hl,18464
238ØØ Ø1 EØ Ø7          ld bc,2Ø16
238Ø3 ED BØ             ldir
238Ø5 Ø6 Ø8             ld b,8
238Ø7 21 ØØ 5Ø          ld hl,2Ø48Ø
2381Ø 11 EØ 48          ld de,18656
L4
23813 C5                push bc
23814 E5                push hl
23815 D5                push de
23816 Ø1 2Ø ØØ          ld bc,32
23819 ED BØ             ldir
23821 D1                pop de
23822 E1                pop hl
23823 C1                pop bc
23824 14                inc d
23825 24                inc h
23826 1Ø F1             djnz L4
23828 11 ØØ 5Ø          ld de,2Ø48Ø
23831 21 2Ø 5Ø          ld hl,2Ø512

23834 Ø1 EØ Ø7          ld bc,2Ø16
23837 ED BØ             ldir
23839 Ø6 Ø8             ld b,8
23841 21 EØ 5Ø          ld hl,2Ø7Ø4
L5
23844 C5                push bc
23845 E5                push hl
23846 Ø6 2Ø             ld b,32
L6
23848 36 ØØ             ld (hl),Ø
2385Ø 23                inc hl
23851 1Ø FB             djnz L6
23853 E1                pop hl
23854 C1                pop bc
23855 24                inc h
23856 1Ø F2             djnz L5
23858 C9                ret
```

Programa 6.13

```
org 2376Ø
2376Ø 21 ØØ 4Ø        ld hl,16384
L18
23763 Ø6 2Ø           ld b,32
L19
23765 7E              ld a,(hl)
23766 F5              push af
23767 23              inc hl
23768 1Ø FB           djnz L19
2377Ø AF              xor a
23771 6F              ld l,a
23772 24              inc h
23773 7C              ld a,h
23774 FE 48           cp 72
23776 2Ø F1           jr nz,L18
23778 11 ØØ 4Ø        ld de,16384
23781 21 2Ø 4Ø        ld hl,16416
23784 Ø1 EØ Ø7        ld bc,2Ø16
23787 ED BØ           ldir
23789 Ø6 Ø8           ld b,8
23791 21 ØØ 48        ld hl,18432
23794 11 EØ 4Ø        ld de,166Ø8
L2Ø
23797 C5              push bc
23798 E5              push hl
23799 D5              push de
238ØØ Ø1 2Ø ØØ        ld bc,32
238Ø3 ED BØ           ldir
238Ø5 D1              pop de
238Ø6 E1              pop hl
238Ø7 C1              pop bc
238Ø8 14              inc d
238Ø9 24              inc h
2381Ø 1Ø F1           djnz L2Ø
23812 11 ØØ 48        ld de,18432
23815 21 2Ø 48        ld hl,18464
23818 Ø1 EØ Ø7        ld bc,2Ø16
23821 ED BØ           ldir
23823 Ø6 Ø8           ld b,8
23825 21 ØØ 5Ø        ld hl,2Ø48Ø
23828 11 EØ 48        ld de,18656
L21
23831 C5              push bc
23832 E5              push hl
23833 D5              push de
23834 Ø1 2Ø ØØ        ld bc,32
23837 ED BØ           ldir
23839 D1              pop de
2384Ø E1              pop hl
23841 C1              pop bc
23842 14              inc d
23843 24              inc h
23844 1Ø F1           djnz L21
23846 11 ØØ 5Ø        ld de,2Ø48Ø
23849 21 2Ø 5Ø        ld hl,2Ø512
23852 Ø1 EØ Ø7        ld bc,2Ø16
23855 ED BØ           ldir
23857 21 FF 57        ld hl,22527
L22
2386Ø Ø6 2Ø           ld b,32
L23
23862 F1              pop af
23863 77              ld (hl),a
23864 2B              dec hl
23865 1Ø FB           djnz L23
23867 3E FF           ld a,255
23869 6F              ld l,a
2387Ø 25              dec h
23871 7C              ld a,h
23872 FE 4F           cp 79
23874 2Ø FØ           jr nz,L22
23876 C9              ret
```

Programa 6.14

```
org 2376Ø
2376Ø 11 FF 57        ld de,22527
23763 21 DF 57        ld hl,22495
23766 Ø1 EØ Ø7        ld bc,2Ø16
23769 ED B8           lddr
23771 Ø6 Ø8           ld b,8
23773 21 EØ 48        ld hl,18656
23776 11 ØØ 5Ø        ld de,2Ø48Ø
L8
23779 C5              push bc
2378Ø E5              push hl
23781 D5              push de
23782 Ø1 2Ø ØØ        ld bc,32
23785 ED BØ           ldir
23787 D1              pop de
23788 E1              pop hl
23789 C1              pop bc
2379Ø 14              inc d
23791 24              inc h
23792 1Ø F1           djnz L8
23794 11 FF 4F        ld de,2Ø479
23797 21 DF 4F        ld hl,2Ø447
238ØØ Ø1 EØ Ø7        ld bc,2Ø16
238Ø3 ED B8           1ddr
238Ø5 Ø6 Ø8           ld b,8
238Ø7 21 EØ 4Ø        ld hl,166Ø8
2381Ø 11 ØØ 48        ld de,18432
L9
23813 C5              push bc
23814 E5              push hl
23815 D5              push de
23816 Ø1 2Ø ØØ        ld bc,32
23819 ED BØ           ldir
23821 D1              pop de
23822 E1              pop hl
23823 C1              pop bc
23824 14              inc d
23825 24              inc h
23826 1Ø F1           djnz L9
23828 11 FF 47        ld de,18431
23831 21 DF 47        ld hl,18399
23834 Ø1 EØ Ø7        ld bc,2Ø16
23837 ED B8           1ddr
23839 21 ØØ 4Ø        ld hl,16384
L1Ø
23842 Ø6 2Ø           ld b,32
L11
23844 36 ØØ           ld (hl),Ø
```

```
23846 23          inc hl          23852 7C        ld a,h
23847 1Ø FB       djnz L11        23853 FE 48     cp 72
23849 AF          xor a           23855 2Ø F1     jr nz,L1Ø
2385Ø 6F          ld l,a          23857 C9        ret
23851 24          inc h
```

Programa 6.15

```
org 2376Ø
2376Ø 21 FF 57    ld hl,22527
L26
23763 Ø6 2Ø       ld b,32
L27
23765 7E          ld a,(hl)
23766 F5          push af
23767 2B          dec hl
23768 1Ø FB       djnz L27
2377Ø 3E FF       ld a,255
23772 6F          ld l,a
23773 25          dec h
23774 7C          ld a,h
23775 FE 4F       cp 79
23777 2Ø FØ       jr nz,L26
23779 11 FF 57    ld de,22527
23782 21 DF 57    ld hl,22495
23785 Ø1 EØ Ø7    ld bc,2Ø16
23788 ED B8       lddr
2379Ø Ø6 Ø8       ld b,8
23792 21 EØ 48    ld hl,18656
23795 11 ØØ 5Ø    ld de,2Ø48Ø
L28
23798 C5          push bc
23799 E5          push hl
238ØØ D5          push de
238Ø1 Ø1 2Ø ØØ    ld bc,32
238Ø4 ED BØ       ldir
238Ø6 D1          pop de
238Ø7 E1          pop hl
238Ø8 C1          pop bc
238Ø9 14          inc d
2381Ø 24          inc h
23811 1Ø F1       djnz L28
23813 11 FF 4F    ld de,2Ø479
23816 21 DF 4F    ld hl,2Ø447
23819 Ø1 EØ Ø7    ld bc,2Ø16
23822 ED B8       lddr
23824 Ø6 Ø8       ld b,8
23826 21 EØ 4Ø    ld hl,166Ø8
23829 11 ØØ 48    ld de,18432
L29
23832 C5          push bc
23833 E5          push hl
23834 D5          push de
23835 Ø1 2Ø ØØ    ld bc,32
23838 ED BØ       ldir
2384Ø D1          pop de
23841 E1          pop hl
23842 C1          pop bc
23843 14          inc d
23844 24          inc h
23845 1Ø F1       djnz L29
23847 11 FF 47    ld de. 18431
2385Ø 21 DF 47    ld hl. 18399
23853 Ø1 EØ Ø7    ld bc. 2016
23856 ED B8       1ddr
23858 21 ØØ 4Ø    ld hl,16384
L3Ø
23861 Ø6 2Ø       ld b,32
L31
23863 F1          pop af
23864 77          ld (hl),a
23865 23          inc hl
23866 1Ø FB       djnz L31
23868 AF          xor a
23869 6F          ld 1,a
2387Ø 24          inc h
23871 7C          ld a,h
23872 FE 48       cp 72
23874 2Ø F1       jn nz L3Ø
23876 C9          ret
```

Programa 6.16

```
   1 REM >?PASO £" BORRAR 65/>?
STEP ?
  5ØØ>DATA 24,79,5,2,"PARAR LA CINTA"
  51Ø DATA Ø,167,6,4, "MOVIMIENTO"
  52Ø DATA Ø,13Ø,2,4, "________"
  525 DATA 24,15,1,2, "PULSE CUALQUIER
TECLA"
  550 DATA 31,170,1,2,"INSTRUCCIONES"
  6ØØ BORDER 4: PAPER 6: INK Ø: CLS
  6Ø5 RESTORE 5ØØ
  61Ø FOR a=1 TO 4
  62Ø INK a: GO SUB 8ØØØ
  63Ø PAUSE 5Ø: NEXT a
  64Ø PAUSE Ø: CLS
  642 INK 2: GO SUB 8ØØØ
  645 PRINT INK Ø; AT 3,Ø; "Este programa
muestra los métodos básicos de movimientos
de primer plano y de fondo."
```

```
 646 PRINT ' "La tabla de abajo muestra las
teclas usadas y sus funciones."
 647 PRINT
 65Ø PRINT "      FONDO"
 655 PRINT " ROLL ; SCROLL
 NAVE ESPACIAL"
 66Ø PRINT "ARRIBA      1   :   Q
     7"
 67Ø PRINT "ABAJO   2 :    W
    8"
 68Ø PRINT "IZQUIERDA   3 :    E
    9"
 69Ø PRINT DERECHA 4:R
    Ø"
 695 PRINT ' "Con la tecla SPACE se vuelve
al BASIC"
 7ØØ RESTORE 525: INK 2: GO SUB 8ØØØ
 7Ø5 IF INKEY$<>"" THEN GO TO 7Ø5
 71Ø IF INKEY$="" THEN GO TO 71Ø
 715 POKE 3ØØ46,Ø: POKE: 3ØØ47,Ø
 72Ø RANDOMIZE USR 3ØØØ1
 725 CLS : RESTORE 73Ø: PAPER 8:
INK Ø: GO SUB 8ØØØ
 73Ø DATA Ø,8Ø,3,1 "ENTER 'GOTO 7
15' para otra vez".
 74Ø STOP
8ØØØ READ x,y,h,w,a$
8Ø1Ø LET c=USR 32393
8Ø2Ø RETURN
9998 CLEAR 3ØØØØ: LOAD "udg" CODE
USR "a": LOAD ""CODE : GO TO 5ØØ
9999 SAVE "MOVEDEMO" LINE 9998: S
AVE "udg" CODE USR "a", 168: SAVE "
M/code "CODE 3ØØØ1,2767
```

Programa 6.17

INSTRUCCIONES

Este programa muestra los métodos básicos de movimiento de primer plano y de fondo.

La tabla de abajo muestra las teclas usadas y sus funciones.

	FONDO		
	ROLL	SCROLL	NAVE ESPACIAL
ARRIBA	1	Q	7
ABAJO	2	W	8
IZQUIERDA	3	E	9
DERECHA	4	R	0

con la Tecla SPACE se vuelve al BASIC

PULSE CUALQUIER TECLA

```
org 40000 30001
30001 F3             di
30002 CD D0 5C       call 23760
30005 18 18          jr COMIENZO
DATA 1
defb 0 0 15 24 49 226 225 224
defb 24 60 255 255 153 255 231
defb 60 0 0 240 24 14071 135 7
COMIENZO
30031 ED 5B 7B 5C    ld de,(23675)
30035 21 37 75       ld hl,DATA 1
30038 01 18 00       ld bc,24
30041 ED B0          ldir
30043 18 0A          jr COMENZAR
DATA2
defb 22 0 0 16 8 17 8 144 145
defb 146
COMENZAR
30055 11 5D 75       ld de,DATA2
30058 01 0A 00       ld bc,10

30061 CD 3C 20       call 8252
30064 11 00 5B       ld de,23296
30067 21 5D 75       ld hl,DATA2
30070 01 0A 00       ld bc,10
30073 ED B0          ldir
MOVER
30075 2A 01 5B       ld hl,(23297)
30078 01 FE EF       ld bc,61438
30081 ED 78          in a,(c)
30083 FE FF          cp 255
30085 28 5D          jr z,LP12
30087 CB 47          bit 0,a

30089 20 01          jr nz,LP1
30091 24             inc h
LP1
30092 CB 4F          bit 1,a
30094 20 01          jr nz, LP2
30096 25             dec h
LP2
3097 CB 57           bit 2,a
30099 20 01          jr nz,LP3


30101 2C             inc l
LP3
30102 CB 5F          bit 3,a
30104 20 01          jr nz, LP4
30106 2D             dec l
LP4
30107 7D             ld a,1
30108 FE 16          cp 22
30110 20 03          jr nz,LP5
30112 2D             dec l
30113 18 05          jr LP6
LP5
30115 FE FF          cp 255
30117 20 01          jr nz,LP6
30119 2C             inc l
LP6
30120 7C             ld a,h
30121 FE 1E          cp 30
30123 20 03          jr nz,LP7
30125 25             dec h
30126 18 05          jr LP8
LP7
30128 FE FF          cp 255
30130 20 01          jr nz,LP8
30132 24             inc h
LP8
30133 ED 5B 01 5B    ld de,(23297)
30137 AF             xor a
30138 E5             push hl
30139 ED 52          sbc hl,de
30141 E1             pop hl
30142 28 24          jr z,LP12
30144 22 01 5B       ld (23297),hl
OVER1
30147 FD 36 57 03    ld (iy+87),3
30151 11 5D 75       ld de,DATA2
30154 01 0A 00       ld bc,10
30157 CD 3C 20       call 8252
30160 11 00 5B       ld de,23296
30163 01 0A 00       ld bc,10
30166 CD 3C 20       call 8252
30169 11 5D 75       ld, de DATA2


3072 21 00 5B        ld hl, 23296
30175 01 0A 00       ld bc,10
30178 ED 80          ldir
LP12
30180 01 FE F7       ld bc,63486
30183 ED 78          in a,(c)
30185 E6 0F          and 15
30187 28 3E          jr z,SCR
30189 FE 03          cp 3
30191 28 3A          jr z,SCR
30193 FE 0C          cp 12
30195 28 36          jr z,SCR
30197 FE 0F          cp 15
30199 28 32          jr z,SCR
30201 F5             push af
30202 FD 36 57 03    ld (iy+87),3
30206 11 5D 75       ld de,DATA2
30209 01 0A 00       ld bc,10
30212 CD 3C 20       call 8252
30215 F1             pop af
30216 CB 47          bit 0,a

30218 F5             push af
30219 CC C9 77       call z,UROLL
30222 F1             pop af
30223 CB 4F          bit 1,a
30225 F5             push af
30226 CC 56 78       call z, DROLL
30229 F1             pop af
30230 CB 57          bit 2,a
30232 F5             push af
30233 CC E3 78       call z,LROLL
30236 F1             pop af
30237 CB 5F          bit 3,a
30239 CC F7 78       call z,RROLL
30242 11 5D 75       ld de,DATA 2
30245 01 0A 00       ld bc,10
30248 CD 3C 20       call 8252
SCR
30251 01 FE FB       ld bc,64510
30254 ED 78          in a,(c)
30256 E6 0F          and 15
30258 28 3E          jr z,DELAY
```

```
30260 FE Ø3          cp 3
30262 28 3A          jr z,DELAY
30264 FE ØC          cp 12
30266 28 36          jr z,DELAY
30268 FE ØF          cp 15
30270 28 32          jr z,DELAY
30272 F5             push af
30273 FD 36 57 Ø3    ld (iy+87),3
30277 11 5D 75       ld de,DATA2
30280 Ø1 ØA ØØ       ld bc,1Ø
30283 CD 3C 2Ø       call 8252
30286 F1             pop af
30287 CB 47          bit Ø,a
30289 F5             push af
30290 CC 8A 76       call z, USCR
30293 F1             pop af
30294 CB 4F          bit 1,a
30296 F5             push af
30297 CC Ø1 77       call z,DSCR
30300 F1             pop af
30301 CB 57          bit 2,a

30303 F5             push af
30304 CC 77 77       call z,LSCR
30307 F1             pop af
30308 CB 5F          bit 3,a
30310 CC AØ 77       call z,RSCR
30313 11 5D 75       ld de,DATA2
30316 Ø1 ØA: ØØ      ld bc,1Ø
30319 CD 3C 2Ø       call 8252
DELAY
30322 21 1Ø 27       ld hl,1ØØØØ
L1
30325 2B             dec hl
30236 7C             ld a,h
30327 B5             or l
30328 2Ø FB          jr nz, L1
30330 Ø1 FE 7F       ld bc,32766
30333 ED 78          in a,(c)
30335 CB 47          bit Ø;a
30337 C2 7B 75       jp nz, MOVE
30340 FD 36 57 ØØ    ld (iy+87),Ø
30344 FB             ei

30345 C9             ret
USCR
30346 11 ØØ 58       ld de,22528
30349 21 2Ø 58       ld hl,2256Ø
30352 Ø1 EØ Ø2       ld bc,736
30355 ED BØ          ldir
30357 Ø6 2Ø          ld b,32
30359 3A 8D 5C       ld a,(23693)
L2
30362 12             ld (de),a
30363 13             inc de
30364 1Ø FC          djnz L2
30366 11 ØØ 4Ø       ld de,16384
30369 21 2Ø 4Ø       ld hl,16416
30372 Ø1 EØ Ø7       ld bc,2Ø16
30375 ED 8Ø          ldir
30377 Ø6 Ø8          ld b,8
30379 21 ØØ 48       ld hl,18432
30382 11 EØ 4Ø       ld de,166Ø8
L3
30385 C5             push bc

30386 E5             push hl
30387 D5             push de
30388 Ø1 2Ø ØØ       ld bc,32
30391 ED BØ          ldir
30393 D1             pop de
30394 E1             pop hl
30395 C1             pop bc
30396 14             inc d
30397 24             inc h
30398 1Ø F1          djnz L3
30400 11 ØØ 48       ld de,18432
30403 21 2Ø 48       ld hl,18464
30406 Ø1 EØ Ø7       ld bc,2Ø16
30409 ED BØ          ldir
30411 Ø6 Ø8          ld b,8
30413 21 ØØ 5Ø       ld hl,2Ø48Ø
30426 11 EØ 4B       ld de,18656
L4
30419 C5             push bc
30420 E5             push hl
30421 D5             push de

30422 Ø1 2Ø ØØ       ld bc,32
30425 ED BØ          ldir
30427 D1             pop de
30428 E1             pop hl
30429 C1             pop bc
30430 14             inc d
30431 24             inc h
30432 1Ø F1          djnz L4
30434 11 ØØ 5Ø       ld de,2Ø48Ø
30437 21 2Ø 5Ø       ld hl,2Ø512
30440 Ø1 EØ Ø7       ld bc,2Ø16
30443 ED BØ          ldir
30445 Ø6 Ø8          ld b,8
30447 21 EØ 5Ø       ld, hl,2Ø7Ø4
L5
30450 C5             push bc
30541 E5             push hl
30452 Ø6 2Ø          ld b,32
L6
30454 36 ØØ          ld (hl),Ø
30456 23             inc hl

30457 1Ø FB          djzn L6
30459 E1             pop hl
30460 C1             pop bc
30461 24             inc h
30462 1Ø F2          dinz L5
30464 C9             ret
DSCR
30465 11 FF 5A       ld de,23295
30468 21 DF 5A       ld hl,23263
30471 Ø1 EØ Ø2       ld bc,736
30474 ED B8          lddr
30476 Ø6 2Ø          ld b,32
30478 3A 8D 5C       ld a,(23693)
L7
30481 12             ld (de),a
30482 13             inc de
30482 1Ø FC          djnz L7
30485 11 FF 57       ld de, 22527
30488 21 DF 57       ld hl,22495
```

```
3Ø491 Ø1 EØ Ø7        ld bc,2Ø16
3Ø494 ED B8           lddr

3Ø496 Ø6 Ø8           ld b,8
3Ø498 21 EØ 48        ld hl,18656
3Ø5Ø1 11 ØØ 5Ø        ld de,2Ø48Ø
L8
3Ø5Ø4 C5              push bc
3Ø5Ø5 E5              push hl
3Ø5Ø6 D5              push de
3Ø5Ø7 Ø1 2Ø ØØ        ld bc,32
3Ø51Ø ED BØ           ldir
3Ø512 D1              pop de
3Ø513 E1              pop hl
3Ø514 C1              pop bc
3Ø515 14              inc d
3Ø516 24              inc h
3Ø517 1Ø F1           djnz L8
3Ø519 11 FF 4F        ld de,2Ø479
3Ø522 21 DF 4F        ld, hl,2Ø447
3Ø525 Ø1 EØ Ø7        ld bc,2Ø16
3Ø528 ED B8           lddr
3Ø53Ø Ø6 Ø8           ld b,8
3Ø552 21 EØ 4Ø        ld hl,166Ø8

3Ø535 11 ØØ 48        ld de,18432
L9
3Ø538 C5              push bc
3Ø539 E5              push hl
3Ø54Ø D5              push de
3Ø541 Ø1 2Ø ØØ        ld bc,32
3Ø544 ED BØ           ldir
3Ø546 D1              pop de
3Ø547 E1              pop hl
3Ø548 C1              pop bc
3Ø549 14              inc d
3Ø55Ø 24              inc h
3Ø551 1Ø F1           djnz L9
3Ø553 11 FF 47        ld de,18431
3Ø556 21 DF 47        ld hl,18399
3Ø559 Ø1 EØ Ø7        ld bc,2Ø16
3Ø562 ED B8           lddr
3Ø564 21 ØØ 4Ø        ld hl,16384
L1Ø
3Ø567 Ø6 2Ø           ld b,32
L11

3Ø569 36 ØØ           ld (hl),Ø
3Ø571 23              inc hl
3Ø572 1Ø FB           djnz L11
3Ø574 AF              xor a
3Ø575 6F              ld l,a
3Ø576 24              inc h
3Ø577 7C              ld a,h
3Ø578 FE 48           cp 72
3Ø58Ø 2Ø F1           jr nz,L1Ø
3Ø582 C9              ret
LSCR
3Ø583 Ø6 18           ld b,24
3Ø585 21 ØØ 58        ld hl,22528
L12
3Ø588 E5              push hl
3Ø589 D1              pop de
3Ø59Ø C5              push bc
3Ø591 23              inc hl
3Ø592 Ø1 1F ØØ        ld bc,31
3Ø595 ED BØ           ldir
3Ø597 3A 8D 5C        ld a, (23693)

3Ø6ØØ 12              ld (de),a
3Ø6Ø1 C1              pop bc
3Ø6Ø2 1Ø FØ           djnz L12
3Ø6Ø4 21 ØØ 4Ø        ld hl,16384
3Ø6Ø7 AF              xor a
3Ø6Ø8 Ø6 CØ           ld b,192
L13
3Ø61Ø C5              push bc
3Ø611 E5              push hl
3Ø612 D1              pop de
3Ø613 23              inc hl
3Ø614 Ø1 1F ØØ        ld bc,31
3Ø617 ED 8Ø           ldir
3Ø619 12              ld (de),a
3Ø62Ø C1              pop bc
3Ø621 1Ø F3           djnz L13
3Ø623 C9              ret
RSCR
3Ø624 Ø6 18           ld b,24
3Ø626 21 FF 5A        ld hl,23295
L14

3Ø629 E5              push hl
3Ø63Ø D1              pop de
3Ø631 C5              push bc
3Ø632 2B              dec hl
3Ø633 Ø1 1F ØØ        ld bc,31
3Ø636 ED B8           lddr
3Ø638 3A 8D 5C        ld a,(23693)
3Ø641 12              ld (de),a
3Ø642 C1              pop bc
3Ø643 1Ø FØ           djnz L14
3Ø645 21 FF 57        ld hl,22527
3Ø648 AF              xor a
3Ø649 Ø6 CØ           ld b,192
L14
3Ø651 C5              push bc
3Ø652 E5              push hl
3Ø653 D1              pop de
3Ø654 2B              dec hl
3Ø655 Ø1 1F ØØ        ld bc,31
3Ø658 ED B8           lddr
3Ø66Ø 12              ld (de),a

3Ø661 C1              pop bc
3Ø662 1Ø DD           djnz L14
3Ø664 C9              ret
UROLL
3Ø665 21 ØØ 58        ld hl,22528
3Ø668 E5              push hl
3Ø669 D1              pop de
3Ø67Ø Ø6 2Ø           ld b,32
L16
3Ø672 7E              ld a,(hl)
3Ø673 F5              push af
3Ø674 23              inc hl
3Ø675 1Ø FB           djnz L16
3Ø677 Ø1 EØ Ø2        ld bc,736
3Ø68Ø ED BØ           ldir
```

```
30682 06 20        ld b,32
L17
30684 F1           pop af
30685 2B           dec hl
30686 77           ld (hl),a
30687 10 FB        djnz L17


30689 21 00 40     ld l,16384
L18
30692 06 20        ld b,32
L19
30694 7E           ld a,(hl)
30695 F5           push af
30696 23           inc hl
30697 10 FB        djnz L19
30699 AF           xor a
30700 6F           ld, la
30701 24           inc h
30702 7C           ld a,h
30703 FE 48        cp 72
30705 20 F1        jr nz,L18
30707 11 00 40     ld de,16384
30710 21 20 40     ld hl,16416
30713 01 E0 07     ld bc,2016
30716 ED B0        ldir
30718 06 08        ld b,8
30720 21 00 48     ld hl,18432
30723 11 E0 40     ld de,16608


L20
30726 C5           push bc
30727 E5           push hl
30728 D5           push de
30729 01 20 00     ld bc,32
30732 ED B0        ldir
30734 D1           pop de
30735 E1           pop hl
30736 C1           pop bc
30737 14           inc d
30738 24           inc h
30739 10 F1        djnz L20
30741 11 00 48     ld de,18432
30744 21 20 48     ld hl,18464
30747 01 E0 07     ld bc,2016
30750 ED B0        ldir
30752 06 08        ld b,8
30754 21 00 50     ld hl,20480
30757 11 E0 48     ld de,18656
L21
30760 C5           push bc


30761 E5           push hl
30762 D5           push de
30763 01 20 00     ld b,32
30766 ED B0        ldir
30768 D1           pop de
30769 E1           pop hl
30770 C1           pop bc
30771 14           inc d
30772 24           inc h
30773 10 F1        djnz L21
30775 11 00 50     ld de,20480
30778 21 20 50     ld hl,20512
```

```
30781 01 E0 07     ld bc,2016
30784 ED B0        ldir
30786 21 FF 57     ld hl,22527
L22
30789 06 20        ld b,32
L23
30791 F1           pop af
30792 77           ld (hl),a
30793 2B           dec hl
30794 10 FB        djnz L23
30796 3E FF        ld a,255
30798 6F           ld l,a
30799 25           dec h
30800 7C           ld a,h
30801 FE 4F        cp 79
30803 20 F0        jr nz,L22
30805 C9           ret
DROLL
30806 21 FF 5A     ld hl,23295
30809 E5           push hl
30810 D1           pop de
30811 06 20        ld b,32
L24
30813 7E           ld a,(hl)
30814 F5           push af
30815 2B           dec hl
30816 10 FB        djnz L24
30818 01 E0 02     ld bc,736
30821 ED B8        lddr
30823 06 20        ld b,32


L25
30825 F1           pop af
30826 23           inc hl
30827 77           ld (hl),a
30828 10 FB        djnz L25
30830 21 FF 57     ld hl,22527
L26
30833 06 20        ld b,32
L27
30835 7E           ld a,(hl)
30836 F5           push af
30837 2B           dec hl
30838 10 FB        djnz L27
30840 3E FF        ld a,255
30842 6F           ld l,a
30843 25           dec h
30844 7C           ld a,h
30845 FE 4F        cp 79
30847 20 F0        jr nz,L26
30849 11 FF 57     ld de,22527
30852 21 DF 57     ld hl, 22495


30855 01 E0 07     ld bc,2016
30858 ED B8        lddr
30860 06 08        ld b,8
30862 21 E0 48     ld hl,18656
30865 11 00 50     ld de,20480
L28
30868 C5           push bc
30869 E5           push hl
30870 D5           push de
30871 01 20 00     ld bc,32
30874 ED B0        ldir
```

```
3Ø876 D1            pop de
3Ø877 E1            pop hl
3Ø878 C1            pop bc
3Ø879 14            inc d
3Ø88Ø 24            inc h
3Ø881 1Ø F1         djnz L28


3Ø883 11 FF 4F      ld de,2Ø479
3Ø886 21 DF 4F      ld hl,2Ø447
3Ø889 Ø1 EØ Ø7      ld bc,2Ø16
3Ø892 ED B8         lddr


3Ø894 Ø6 Ø8         ld b,8
3Ø896 21 EØ 4Ø      ld hl,166Ø8
3Ø899 11 ØØ 48      ld de,18432
L29
3Ø9Ø2 C5            push bc
3Ø9Ø3 E5            push hl
3Ø9Ø4 D5            push de
3Ø9Ø5 Ø1 2Ø ØØ      ld bc,32
3Ø9Ø8 ED BØ         ldir
3Ø91Ø D1            pop de
3Ø911 E1            pop hl
3Ø912 C1            pop bc
3Ø913 14            inc d
3Ø914 24            inc h
30915 1Ø F1         djnz L29
3Ø917 11 FF 47      ld de,18431
3Ø92Ø 21 DF 47      ld hl,18399
3Ø923 Ø1 EØ Ø7      ld bc,2Ø16
3Ø926 ED B8         lddr
3Ø928 21 ØØ 4Ø      ld hl,16384
L3Ø


3Ø931 Ø6 2Ø         ld b,32
L31
3Ø933 F1            pop af
3Ø934 77            ld (hl),a
3Ø935 23            inc hl
3Ø936 1Ø FB         djnz L31
3Ø938 AF            xor a
3Ø939 6F            ld l,a
3Ø94Ø 24            inch h
3Ø941 7C            ld a,h
3Ø942 FE 48         cp 72
3Ø944 2Ø F1         jr nz,L3Ø
3Ø946 C9            ret
LROLL
3Ø947 Ø6 D8         ld b,216
3Ø949 21 ØØ 4Ø      ld hl,16384
L32
3Ø952 E5            push hl
3Ø953 D1            pop de
3Ø954 C5            push bc
3Ø955 7E            ld a,(hl)


3Ø956 23            inc hl
3Ø957 Ø1 1F ØØ      ld bc,31
3Ø96Ø ED BØ         ldir
3Ø962 12            ld (de),a
3Ø963 C1            pop bc
3Ø964 1Ø F2         djnz L32
3Ø966 C9            ret
RROLL
3Ø967 Ø6 D8         ld b,216
3Ø969 21 FF 5A      ld hl,23295
L33
3Ø972 E5            push hl
3Ø973 D1            pop de
3Ø974 C5            push bc
3Ø975 7E            ld a,(hl)
3Ø976 2B            dec hl
3Ø977 Ø1 1F ØØ      ld bc,31
3Ø98Ø ED B8         lddr
3Ø982 12            ld (de),a
3Ø983 C1            pop bc
3Ø984 1Ø F          djnz L33


3Ø986 C9            ret
```

Programa 6.18

Observe que en estas rutinas se hace uso extensivo de las instrucciones LDIR y LDDR de transferencia de bloque, y que después de estas instrucciones los registros DE y HL contienen la dirección final + 1 (LDIR) o − 1 (LDDR). Se utiliza este hecho para volver a cargar HL o DE con una dirección específica para posteriores rutinas del programa. Por ejemplo, ATTR. UP SCROLL (programa 6.1 B) utiliza el hecho de que después de LDIR el registro DE contiene la dirección del atributo "START of LINE 24" y de esta forma no es necesario volver a cargar DE con esta dirección. Los atributos actuales de la línea 24 son reemplazados por los atributos permanentes de pantalla guardados en la variable ATTR P (23693) del sistema.

Para mostrar cada tipo de movimiento he escrito el programa

6.17 que viene en la cinta "MOVEDEMO". Como el programa tiene casi 1000 bytes de largo no es práctico guardarlo todo en una sentencia REM (ocuparía más de una pantalla completa), entonces lo he escrito para utilizarlo por encima de RAMTOP comenzando en la dirección 30001 (programa 6.18).

El código máquina en la sentencia REM se toma del programa de números aleatorios (programa 4.1) y es utilizado para establecer el fondo de pantalla. El programa es auto-explicativo y hace llamadas a diversas rutinas, dependiendo de las teclas que se pulsan. Los atributos cambian al mismo tiempo que los caracteres que se visualizan. Usted puede mover el fondo y la nave diagonalmente y si se pulsan dos teclas opuestas, entonces la rutina es pasada por alto. Pulsando "SPACE" se vuelve al BASIC. Observe que el movimiento vertical es más bien desigual y no es realmente apropiado debido a la transición de una zona de pantalla a otra. De hecho, los programas ponen la línea 1 en la línea 8 momentáneamente antes de desplazar la línea 9 hacia la línea 8.

Los programas de pantalla UP/DOWN-SCROLL y ROLL (programas de 6.13 a 6.16) fueron escritos de forma que puedan ser fáciles de seguir por usted sin subrutinas para hacerlos dependientes de la posición. Para conseguir un movimiento más suave estas rutinas deben abordarse de forma diferente, esto es, sacando una fila de una vez, en vez de sacar la línea superior de cada columna, luego la segunda línea de cada fila, etc., como se ha hecho hasta ahora. Para mostrar como se consigue esto he reescrito el programa SCREEN UP SCROLL (programa 6.19A). Como puede ver contiene una subrutina y una sub-subrutina, y es bastante más complicado. Quizás quiera usted reescribir la rutina SCREEN DOWN SCROLL de la misma forma, como ejercicio para estar seguro de que comprende completamente como funciona.

```
org 2376Ø
2376Ø 21 ØØ 4Ø        ld hl,16384
23763 CD Ø1 5D        call SECT
23766 EB              ex de,hl
23767 21 ØØ 48        ld hl,18432
2377Ø E5              push hl
23771 CD 15 5D        call UP
23774 E1              pop hl
23775 CD Ø1 5D        call SECT
23778 EB              ex de,hl
23779 21 ØØ 5Ø        ld hl,2Ø48Ø
23782 E5              push hl
23783 CD 15 5D        call UP
23786 E1              pop hl
23787 CD Ø1 5D        call SECT
2379Ø 21 EØ 5Ø        ld hl,2Ø7Ø4
L4
23793 E5              push hl
23794 Ø6 2Ø           ld b,32
L3
23796 36 ØØ           ld (hl),Ø
23798 23              inc hl
23799 1Ø FB           djnz L3
238Ø1 E1              pop hl
238Ø2 24              inc h
238Ø3 7C              ld a,h
238Ø4 FE 58           cp 88
238Ø6 2Ø F1           jr nz,L4
```

```
23808 C9          ret
SECT
23809 06 07       ld b,7
L1
23811 C5          push bc
23812 01 20 00    ld bc,32
23815 E5          push hl
23816 D1          pop de
23817 09          add hl,bc
23818 E5          push hl
23819 D5          push de
23820 CD 15 5D    call UP
23823 D1          pop de

23824 E1          pop hl
23825 C1          pop bc
23826 10 EF       djnz L1
23828 C9          ret
UP
23829 06 08       ld b,8
L2
23831 C5          push bc
23832 E5          push hl
23833 D5          push de
23834 01 20 00    ld bc,32
23837 ED B0       ldir
23839 D1          pop de
23840 E1          pop hl

23841 C1          pop bc
23842 14          inc d
23843 24          inc h
23844 10 F1       djnz L2
23846 C9          ret
```

Programa 6.19A

Si compara los programas 6.13 y 6.19A verá que hemos reducido la memoria requerida de 99 bytes a 87 bytes, pero aún no es tan compacto como podría serlo. Para mostrar cómo podríamos reducir la memoria y el tiempo de ejecución todavía más y obtener una característica útil extra consideraremos el programa 6.14, UP ROLL, que tal como está toma 117 bytes. Hasta ahora el programa maneja cada fila de carácter línea por línea y tiene el “fallo” de que tiene que transferir temporalmente la fila 1 a la fila 8 y la fila 9 a la fila 17. Podemos reescribirlo sobre las líneas del programa 6.21 para superar este problema, pero en vez de ello abordaremos el asunto con un enfoque completamente diferente para obtener un programa de sólo 76 bytes de largo.

En vez de manejar la pantalla con una fila cada vez, podemos escribir un programa que saque una columna cada vez. Entonces la rutina debe repetirse para las 32 columnas. De esta forma seremos capaces de seleccionar la columna que queramos listar, ya sea en bloques o una a una, y lo que es más, no requeriremos 256 bytes de memoria para almacenar la fila superior de la pantalla como en el programa 6.14, pues en vez de ello emplearemos sólo ocho bytes para el carácter de la columna superior. El programa 6.19B muestra como se realiza esta reducción de memoria.

```
org 23760
CAPITULO 6
PROGRAMA 19B

LA PANTALLA PRESENTA 'n' COLUMNAS

EJEMPLO

PRESENTAR DESDE LAS COMUMNAS 5 A
14 INC
23760 AF          xor a

LD DE,16383+COLUMNA DE COMIENZO

1era COLUMNA = 1
```

```
23761 11 Ø4 4Ø          ld de,16388

LD B,Nu. de columnas

23764 Ø6 ØA             ld b,1Ø
L5
23766 D5                push de
23767 C5                push bc

GUARDAR EL CARACTER DE LA LINEA
SUPERIOR EN LA PILA

23768 D5                push de
23769 E1                pop hl
2377Ø Ø6 Ø8             ld b,8
L6
23772 7E                ld a,(hl)
23773 F5                push af
23774 24                inc h
23775 1Ø FB             djnz L6

TRANSFERIR 7 LINEAS

L4
23777 Ø6 Ø7             ld b,7
L2

23779 C5                push bc
2378Ø 21 2Ø ØØ          ld hl,32
23783 19                add hl,de
23784 E5                push hl

TRANSFERIR 8 BYTES

23785 Ø6 Ø8             ld b,8
L1
23787 7E                ld a,(hl)
23788 12                ld (de),a
23789 24                inc h

2379Ø 14                inc d
23791 1Ø FA             djnz L1
23793 D1                pop de
23794 C1                pop bc
23795 1Ø EE             djnz L2

TRANSFERIR ZONA DE PANTALLA

23797 21 2Ø Ø7          ld hl,1824

238ØØ 19                add hl,de
238Ø1 7C                ld a,h

EXAMINAR SI ESTA FUERA DE PANTALLA
por ejemplo COMIENZO DE ATRIBUTOS

238Ø2 FE 58             cp 88
238Ø4 28 ØC             jr z,END
238Ø6 E5                push hl
238Ø7 Ø6 Ø8             ld b,8
L3
238Ø9 7E                ld a,(hl)
2381Ø 12                ld (de),a
23811 24                inc h
23812 14                inc d
23813 1Ø FA             djnz L3
23815 D1                pop de
23816 18 D7             jr L4

SACAR CARACTER DE LA PILA
Y PONERLO EN LA FILA INFERIOR DE
LA PANTALLA

FIN
23818 11 2Ø ØØ          ld de,32
23821 ED 52             sbc hl,de
23823 Ø6 Ø8             ld b,8
L7
23825 F1                pop af
23826 77                ld (hl),a
23827 25                dec h
23828 1Ø FB             djnz L7

MOVERSE A LA SIGUIENTE COLUMNA
Y REPETIR

2383Ø C1                pop bc
23831 D1                pop de
23832 13                inc de
23833 1Ø BB             djnz L5
23835 C9                ret
```

Programa 6.19B

Observará que empezamos el programa 6.19B cargando el par de registro DE con la dirección del byte superior de la columna de comienzo, y el registro B con el número de columnas a listar. El programa es completamente auto-explicativo, y puede posicionarse en cualquier parte de la memoria.

Aún no hemos terminado de ahorrar memoria. ¿Podría creer que es posible reducir el programa 6.19B a 58 bytes y añadir una facilidad extra para poder definir las filas requeridas, y

conseguir una rutina que visualice una ventana de líneas? Pues bien, el programa 6.19C hace justamente eso.

Se ahorra memoria haciendo un examen, cuando se llega al final de una zona de pantalla, por medio de la instrucción BIT 0,H. Si BIT 0,H no es cero entonces tenemos que ir hasta el final de una zona, y en vez de sumar 32d a DE para mover hacia arriba una línea dentro de una zona, sumamos 1824 para movernos hasta la "primera" línea de la siguiente zona. Utilizando este método podemos definir cuantas filas requieren desplazarse, mediante el número de veces que se repite el lazo L4. Al principio se definen ROW/COLUMN con la dirección de visualización contenida en el par de registros DE (23671/3). Para calcular esta dirección de comienzo utilice la siguiente fórmula:

$$16384 + (1824 * \text{INT}(\text{ROW}/8)) = ((\text{ROW} - \text{INT}(\text{ROW}/8)) * 32) + \text{COLUMN}$$

En esta fórmula se emplean los números ROW/COLUMN de la Spectrum, es decir 1.ª fila = 0, 1.ª columna = 0.

Ejemplo: ROW 17 COLUMN 5

$$= 16384 + (1824 * 2) + ((17 - 6) * 32) + 5$$
$$= 16384 + 3648 + 32 + 5$$
$$= 20069$$

```
org 2376Ø
CAPITULO 6
PROGRAMA 19C

VENTANA RODANTE HACIA ARRIBA

EJEMPLO
COLUMNAS 4 A 1Ø
FILAS 5 A 18

2376Ø AF                  xor a

LD DE,DIRECCION COMIENZO PANTALLA
por ejemplo DIRECCION COMIENZO
COLUMNA/FILA

23761 11 83 4Ø            ld de,16515

LD B,N. de columnas

23764 Ø6 Ø7               ld b,7

L6
23766 D5                  push de
23767 C5                  push bc
23768 D5                  push de
23769 E1                  pop hl
2377Ø Ø6 Ø8               ld b,8
L1
23772 7E                  ld a,(hl)
23773 F5                  push af
23774 24                  inc h
23775 1Ø FB               djnz L1

LD B,N. de filas-1!

23777 Ø6 ØD               ld b,13
L4
23779 C5                  push bc
2378Ø 21 2Ø ØØ            ld hl,32
23783 19                  add hl,de

EXAMINAR CAMBIO DE ZONA
DE PANTALLA

23784 CB 44               bit Ø,h
23786 28 Ø4               jr z,L2
23788 21 2Ø Ø7            ld hl,1824
23791 19                  add hl,de
L2
23792 E5                  push hl
23793 Ø6 Ø8               ld b,8
L3
23795 7E                  ld a,(hl)
```

```
23796 12        ld (de),a       23808 F1        pop af
23797 14        inc d
23798 24        inc h           23809 77        ld (hl),a
23799 10 FA     djnz L3         23810 10 FB     djnz L5
23801 D1        pop de          23812 C1        pop bc
23802 C1        pop bc          23813 D1        pop de
23803 10 F6     djnz L4         23814 D1        pop de
23805 06 08     ld b,8          23815 13        inc de
L5                              23816 10 CC     djnz,L6
23807 25        dec h           23818 C9        ret
```

Programa 6.19C

Para visualizar todas las líneas de la pantalla DE debe contener 16384, el registro B (antes de L6) debe contener 32 y el registro B (antes de L4) debe contener 23. Este programa se puede adaptar fácilmente para obtener las rutinas UP/DOWN SCROLL y DOWN ROLL.

Movimiento de pixel

Este tipo de movimiento de caracteres de primer plano no es realmente aconsejable para programas de juegos, ya que suele ser más bien lento. El movimiento se reduce en un factor de ocho si, por ejemplo, trata de ver si un carácter golpea a un obstáculo; entonces cada bit de la forma del carácter debe ser examinado utilizando el equivalente de POINT x,y en código máquina para ver si su nueva posición está a cero antes de volver a dibujar. Así, si un carácter tiene un tamaño de 16 × 16 pixels, entonces aunque sólo se examinen los pixels de las posiciones externas habría que hacer cerca de 100 llamadas a la rutina POINT antes de mover el carácter. Todo ello requiere tiempo, y por lo tanto la velocidad del movimiento se reduce drásticamente. Sin embargo, en el movimiento de fondo es posible mover un pixel cada vez mientras los atributos permanezcan fijos (si no es así no se puede mover un pixel).

El programa de demostración "PIXEL DEMO" se dispone en cinta y es nuestro primer programa de juegos de reflejos. Demuestra la lentitud del movimiento de pixel y también el hecho de que hay muchos parpadeos debido al movimiento del fondo. El balón debe ser sacado de la pantalla durante el movimiento de fondo y se debe hacer una prueba mediante POINT antes de volver a dibujarlo. Este tiempo de retardo es perceptible pero se ha conseguido hacerlo lo más corto posible mediante la elección del balón, para el cual sólo se necesita la línea externa; de esta forma reducimos al mínimo el número de veces que hay que examinar POINT y el número de llamadas a PLOT. Los

principios de actuación son los mismos que en el movimiento de cuadros de carácter, pues las posiciones x,y de PLOT se almacenan como datos, se actualiza y examina un conjunto duplicado de datos actualizados, se borra el balón utilizando OVER 1, y se vuelve a dibujar utilizando los datos duplicados. Observe que el balón puede moverse fuera de la parte izquierda y derecha de la pantalla en forma de enrosque. De hecho esto es necesario para eludir las flechas de la línea de abajo.

Para aquellos con aparatos en color hay que indicar que el programa está en blanco y negro porque si se empleasen más de dos colores, el balón podría cambiar de color al moverse de un atributo de cuadro a otro; quizás esto no importe, porque también sale bien en un aparato en blanco y negro.

El programa muestra los principios anteriores de presentación de pantalla rodante hacia la izquierda y hacia la derecha de forma que cada carácter es rotado a la izquierda con RLA, o hacia la derecha con RRA y, si se transfiere un bit al CARRY (acarreo) este es recogido por el byte del carácter adyacente. Este se tratará de nuevo en el Capítulo 9.

Para mostrar el rodamiento descendente con pixel, el siguiente programa (programa 6.20) dibuja una onda sinusoidal y una línea en la parte superior de la pantalla; después la rodará hacia abajo hasta que terminamos con una "carretera" negra suavemente ondulada. Esto puede utilizarse en un programa de carreras de coches. También muestra la forma de poner "steps" no enteros en un lazo (ver Capítulo 7).

```
 10 FOR a=0 TO 4*PI STEP PI/44
 PLOT 100+SIN a*25,175: DRAW 100,0
  30 REM LLAMADA DESPLAZAMIENTO
HACIA ABAJO DE UN PIXEL

 40 NEXT a

org 23760

SET a=0 ( en la calculadora )

23760 EF              rst 40
defb 160 56

GUARDAR valor a en DATA

L1
23763 11 35 5D        ld de,DATA
23766 01 05 00        ld bc,5
23769 ED B0           ldir

PILA 4*PI ( en calc )
23771 EF              rst 40
defb 163 56
23774 34              inc (hl)
23775 34              inc (hl)
23776 34              inc (hl)

SUB a-4*PI

23777 EF              rst 40
defb 3 56

EXAMINAR 1er BYTE DE MANTISA
PARA -ve RESPUESTA

23780 23              inc hl
23781 7E              ld a,(hl)
23782 CB 7E           bit 7,a
23784 20 04           jr nz,L2

BORRAR CALCULADORA si -ve

23786 CD F1 2B        call 11249

VOLVER AL BASIC

23789 C9              ret

BORRAR CALCULADORA

L2
23790 CD F1 2B        call 11249
23793 CD 3A 5D        call ROLL
```

```
APILAR DATO a

23796 CD 24 5D          call PILA

100+SIN a*25=x

23799 EF                rst 40
sine
defb 31
stack 25
defb 52 53 72
mult
defb 4
stack 100

defb 52 55 72
add
defb 15
stack 175 (=y)
defb 52 56 47 56

LLAMADA PLOT x,y

23813 CD DC 22          call 8924

GUARDAR H'L'

23816 D9                exx
23817 E5                push hl
23818 D9                exx

LD B,0  LD C,100

23819 01 64 00          ld bc,100

LD DE, +ve +ve

23822 11 01 01          ld de,257

LLAMADA DIBUJAR

23825 CD BA 24          call 9402

RECUPERAR H'L'

23828 D9                exx
23829 E1                pop hl
23830 D9                exx

APILAR DATO a

23831 CD 24 5D          call PILA

a+PI/44
23834 EF                rst 40
stack PI/2
defb 163

stack 22
defb 52 53 48
divide
defb 5
add
defb 15
end calc routine
defb 56
23842 18 AF             jr L1
```

```
APILAR un SUB

PILA
23844 21 35 5D          ld hl,DATA
23847 ED 5B 65 5C       ld de,(23653)
23851 01 05 00          ld bc,5
23854 ED B0             ldir
23856 ED 53 65 5C       ld (23653),de
23860 C9                ret

GUARDAR DATO

DATA
defb 0 0 0 0 0

DESPLAZAR HACIA ABAJO

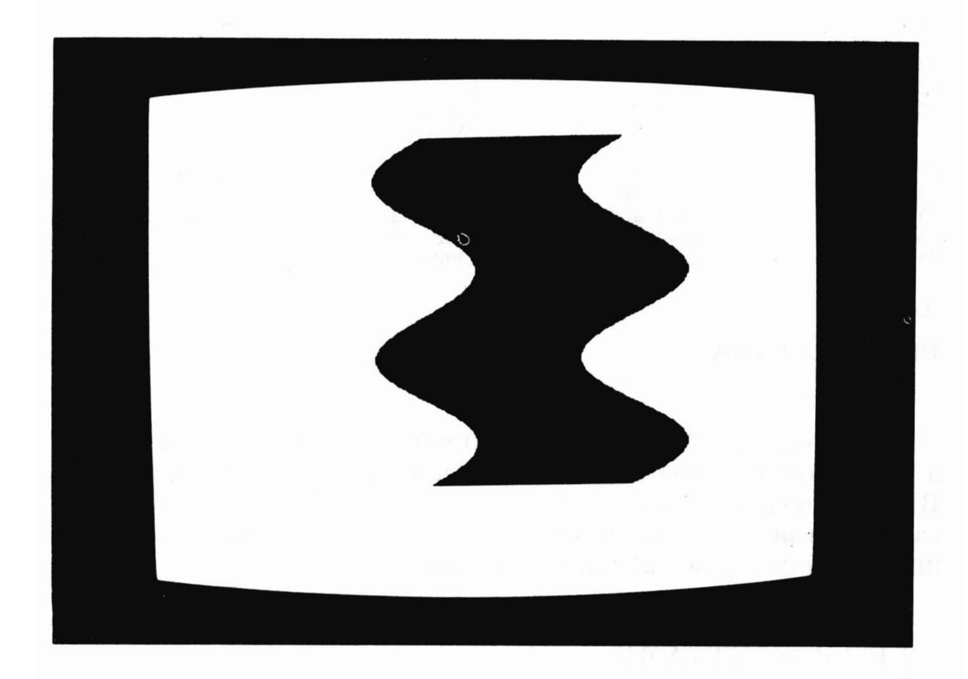
DESPLAZAR
23866 06 20             ld b,32
23868 21 A0 57          ld hl,22432
L3
23871 7E                ld a,(hl)
23872 F5                push af
23873 23                inc hl
23874 10 FB             djnz L3
23876 21 BF 56          ld hl,22207
23879 11 BF 57          ld de,22463
23882 06 06             ld b,6
23884 CD 86 5D          call ABAJO
23887 21 E0 4F          ld hl,20448
23890 11 00 50          ld de,20480
23893 01 20 00          ld bc,32


23896 ED B0             ldir
23898 21 FF 4E          ld hl,20223
23901 11 FF 4F          ld de,20479
23904 06 08             ld b,8
23906 CD 86 5D          call ABAJO
23909 21 E0 47          ld hl,18400
23912 11 00 48          ld de,18432
23915 01 20 00          ld bc,32
23918 ED B0             ldir
23920 21 FF 46          ld hl,18175
23923 11 FF 47          ld de,18431
23926 06 08             ld b,8
23928 CD 86 5D          call ABAJO
23931 06 20             ld b,32
23933 21 1F 40          ld hl,16415
L4
23936 F1                pop af
23937 77                ld (hl),a
23938 2B                dec hl
23939 10 FB             djnz L4
23941 C9                ret


ABAJO
23942 C5                push bc
23943 06 07             ld b,7
L5
23945 C5                push bc
23946 E5                push hl
23947 E5                push hl
23948 01 20 00          ld bc,32
23951 ED B8             lddr
```

```
23953 D1          pop de        23967 E5          push hl
23954 E1          pop hl
23955 25          dec h         23968 E5          push hl
23956 C1          pop bc        23969 Ø1 2Ø ØØ    ld bc,32
23957 1Ø F2       djnz L5       23972 ED B8       lddr
23959 7C          ld a,h        23974 E1          pop hl
2396Ø C6 Ø8       add a,8       23975 D1          pop de
23962 67          ld h,a        23976 25          dec h
23963 7D          ld a,l        23977 C1          pop bc
23964 D6 2Ø       sub 32        23978 1Ø DA       djnz ABAJO
23966 6F          ld l,a        2398Ø C9          ret
```

Programa 6.20



La rutina PIXEL DOWN ROLL que hay al final del programa 6.20 ha sido escrita de nuevo para que usted pueda ver fácilmente los procedimientos necesarios. Puede reescribirse en las líneas del programa 6.19C para hacer rodar en forma descendente con pixel una ventana, convirtiéndolo al mismo tiempo en "independiente de la posición" y usando menos memoria. En su forma actual utiliza 115 bytes (+ 32 bytes para almacenar la línea inferior), pero reescrita como el programa 6.20A utiliza sólo 44 bytes (+ 1 para almacenar el byte de la columna inferior).

```
org 23760
CAPITULO 6
PROGRAMA 20A

VENTANA RODANTE HACIA ABAJO

EJEMPLO

COLUMNAS 4 A 10
LINEAS 4 A 39

REM LINEA 1/COLUMNA 1=PARTE
SUPERIOR LH.

23760 11 83 46      ld de,18051

LD B, Columnas

23763 06 07         ld b,7
L2
23765 C5            push bc


23766 D5            push de
23767 1A            ld a,(de)
23768 F5            push af
23769 D5            push de
23770 E1            pop hl

LDB, Líneas-1¡
23771 25            dec h
23772 06 23         ld b,35
L1
23774 E5            push hl
23775 7E            ld a,(hl)
23776 12            ld (de),a
23777 D1            pop de
23778 7C            ld a,h
23779 25            dec h
23780 E6 07         and 7
23782 20 0A         jr nz,L3
23784 7D            ld a,l
23785 D6 20         sub 32

23787 6F            ld l,a
23788 38 04         jr c,L3
23790 7C            ld a,h
23791 C6 08         add a,8
23793 67            ld h,a
L3
23794 10 EA         djnz L1
23796 F1            pop af
23797 12            ld (de),a
23798 D1            pop de
23799 C1            pop bc
23800 13            inc de
23801 10 DA         djnz L2
23803 C9            ret
```

Programa 6.20A

En el registro B se guarda el número de columnas que se van a hacer rodar (antes de L2) y también se guarda en el registro B el número de líneas − 1 (antes de L1). El registro DE se carga con la dirección de la pantalla del byte de la primera columna inferior y se calcula utilizando:

LET A = LINE (0 es la primera línea)
LET B = INT (A/64)
LET C = A − (b * 64)
LET D = INT (C/8)
LET E = C − (D * 8)
LET F = COLUMN (0 es la primera línea)
ADDRESS = 16384 + (1824 * B) + (256 * E) + (32 * (D − 1)) + F
EXAMPLE LINE 38 : COLUMN 3
A = 38 : B = 0 : C = 38 : D = 3 : E = 6 : F = 3
ADDRESS = 16384 + (1824 * 0) + (256 * 6) + (32 * 4) + 3 = 18051

Como esta fórmula es más bien larga, puede escribir un programa en código máquina para que haga el cálculo. Empiece guardando el número LINE/COLUMN en el par de registros HL y termine con la dirección de pantalla en DE para que así

pueda ir directamente a la rutina ROLL (alternativamente puede tener una tabla que muestre las direcciones individuales de las líneas/columnas para el mapa de memoria de la pantalla).

¿Por qué no intenta escribir tal rutina? Podría serle útil para muchos programas, por ejemplo, para sacar caracteres directamente en la pantalla con una instrucción POKE. Si se atasca, el programa 6.20B le ofrece un método, y utiliza sólo 29 bytes (serían 28 bytes si LINE/COLUMNE se hubiesen almacenado en HL utilizando LD HL, 034Fh). La he unido al programa 6.20A.

Estudie este programa hasta que crea que sabe como funciona, y conseguirá tener mejor conocimiento del fichero de visualización.

```
org 2376Ø
CAPITULO 6
PROGRAMA 2ØB

VENTANA RODANTE HACIA ABAJO

EJEMPLO

COLUMNAS 3 A 9
LINEAS 56 A 79

REM LINEA Ø/COLUMNA Ø=PARTE
SUPERIOR LH.

LD H, Línea

2376Ø 26 4F        ld h,79

LD L,Columna

23762 2E Ø3        ld l,3


23764 11 ØØ 4Ø     ld de,16384
23767 7C           ld a,h
23768 E6 Ø7        and 7
2377Ø B2           or d
23771 57           ld d,a
23772 7C           ld a,h
23773 E6 CØ        and 192
23775 1F           rra
23776 1F           rra
23777 1F           rra
23778 82           add a,d
23779 57           ld d,a
2378Ø 7C           ld a,h
23781 E6 38        and 56
23783 17           rla
23784 17           rla
23785 5F           ld e,a
23786 7D           ld a,l
23787 83           add a,e
23788 5F           ld e,a

LD B, Columnas

23789 Ø6 Ø7        ld b,7
L2
23791 C5           push bc
23792 D5           push de
23793 1A           ld a,(de)
23794 F5           push af
23795 D5           push de
23796 E1           pop hl
23797 25           dec g

LD B, Lineas-1 !

23798 Ø6 17        ld b,23
L1
238ØØ E5           push hl
238Ø1 7E           ld a,(hl)
238Ø2 12           ld (de),a
238Ø3 D1           pop de
238Ø4 7C           ld a,h

238Ø5 25           dec h
238Ø6 E6 Ø7        and 7
238Ø8 2Ø ØA        jr nz,L3
2381Ø 7D           ld a,l
23811 D6 2Ø        sub 32
23813 6F           ld l,a
23814 38 Ø4        jr c,L3
23816 7C           ld a,h

23817 C6 Ø8        add a,8
23819 67           ld h,a
L3
2382Ø 1Ø EA        djnz L1
23822 F1           pop af
23823 12           ld (de),a
23824 D1           pop de
23825 C1           pop bc
23826 13           inc de
23827 1Ø DA        djnz L12
23829 C9           ret
```

Programa 6.20B

Estos dos programas para el movimiento de pixel (programa 6.20 y 6.20B) muestran los métodos de rodamiento del fondo a izquierda/derecha y hacia abajo. Le dejo a usted que haga programas para UP ROLL (rodar hacia arriba) y SCROLL. Debe recordarse que en código máquina no hay ningún método perfecto. Cada persona tiene su propia forma particular de hacer las cosas, de escoger los registros, y de decidir si utiliza instrucciones "push/pop" o "store" para guardar un valor, etc., y tan pronto como el programa funcione de acuerdo con lo que uno desea entonces está bien. Muchos de los programas hasta ahora tratados podrían volverse a escribir para ahorrar memoria o reducir el tiempo de ejecución, pero entonces no serían tan fácil de seguir por un programador principiante.

Hay una rutina en ROM más que es útil y que no se dispone en el BASIC, que es CROLL n líneas con atributos, donde n debe ser mayor o igual a dos (pero no mayor que 24, por supuesto).

El programa 6.21 muestra la rutina que se está usando para borrar las líneas desde la 16 a la 24 inclusive (nueve líneas) llamando a la rutina nueve veces. Observe que el número de líneas que se van a visualizar se cuenta desde la línea 24. Puede usarse esta rutina en vez de la nuestra, escrita en el programa "MOVEDEMO" para el usuario. El registro contiene el número de líneas − 1 que se van a sacar, y después se hace una llamada a la 3584. La línea inferior se sobreimprime con espacios utilizando colores de ATTRP.

```
org 23760
23760 06 009          ld b,9
L1
23762 C5              push bc
23763 06 08           ld b,8
23765 CD 00 0E        call 3584
23768 C1              pop bc
23769 10 F7           djnz L1
23771 C9              ret
```

Programa 6.21

Finalmente mostramos una rutina para desplazar una pantalla a la izquierda (o derecha) medio carácter cada vez. Este programa, aunque corto, es bastante difícil de entender, y utiliza la instrucción de rotación decimal a la izquierda (o rotación decimal a la derecha). Esta instrucción lleva consigo el manejo de nibbles del registro A y que los contenidos de la dirección se coloquen en el par de registros HL. RLD transfiere los bits 0 a 3

(HL) a los bits 7 a 4 (HL); los bits 7 a 4 (HL) a los bits 0 a 3,A; y los bits 0 a 3,A a los bits 0 a 3 (HL). No afecta a ninguno de los "flags" (bits de estado). En el programa 6.22 puede ver que se utiliza esta instrucción en cada línea de pantalla empleando el registro A para almacenar el nibble de la derecha y para colocarlo en el siguiente carácter del nibble de la izquierda. El primer carácter del nibble de la izquierda se queda en blanco porque el registro A contiene un cero al comienzo de cada línea. Al final de cada línea, el registro A contiene el nibble de la izquierda que será desplazado fuera de la pantalla. Esto se utiliza para sumarlo al byte de comienzo de línea para producir un efecto de enrosque. Si quiere un cambio de fondo continuo puede guardar el nibble de reemplazamiento de la derecha en forma de "data" para sumarlo al byte de la derecha, en vez de utilizar el nibble desplazado de la pantalla, como antes se ha indicado. Dejaré que usted escriba la rutina RIGHT ROLL (al comienzo del fichero de visualización hay una indicación–).

```
org 23760
CAPITULO 6
PROGRAMA 22

RUTINA DE DESPLAZAMIENTO A IZQ.
DE MEDIO CUADRO DE CARACTER

23760 F3                di

Empezar al final de la pantalla

23761 21 FF 57          ld hl,22527

N.° de líneas de pantalla

23764 06 C4             ld b,196
L1
23766 C5                push bc

Reinicializar a=0

23767 AF                xor a

N.° de bytes/línea

23768 06 20             ld b,32

Guardar comienzo de línea

23770 E5                push hl

Instrucción rodar a izquierda
un nibble a (hl)

L2
23771 ED 6F             rld
23773 2B                dec hl
23774 10 FB             djnz L2

Obtener comienzo de línea en de

23776 D1                pop de

Guardar actual pos. de pant.

23777 EB                ex de, hl

Sumar nibble LH
a byte RH

23778 86                add a,(hl)
23779 77                ld (hl),a

Recuperar pos. de pant. en hl

23780 EB                ex de,hl
23781 C1                pop bc
23782 10 EE             djnz L1
23784 FB                ei
23785 C9                ret
```

Programa 6.22

7 MUSICA Y EFECTOS DE SONIDO

En este capítulo estudiaremos las formas de mejorar sus gráficos animados con sonidos interesantes y "música". Pero primero debemos echar una ojeada a la forma que tiene la Spectrum de almacenar números en forma de coma flotante, y también veremos cómo convertir un número en forma de coma flotante en su forma comprimida. Si sus matemáticas están abandonadas no se preocupe demasiado por la teoría, porque la Spectrum puede ayudarle. Se trata sencillamente de que cualquier número, decimal, entero, positivo o negativo puede expresarse en forma de cinco-bytes utilizando un procedimiento de conjunto para definir cada byte. Al primer byte se le llama *exponente*, y a los cuatro restantes *mantisa*.

La conversión de un número "normal" a su forma de cinco-bytes requiere los cuatro pasos siguientes:

1. Expresar el número en su forma *binaria*.
   Por ejemplo 11.375 sería 1011.011
   Recuerde que 0.1 en binario = $^1/_2$
   0.01 en binario = $^1/_4$
   etc.
2. Calcular el *exponente*.
   Mover la coma decimal a la izquierda o a la derecha hasta que esté a la izquierda del primer 1 en el número binario, y contar el número de movimientos realizados.
   El EXPONENTE será 128 + el número de movimientos realizados.
   Si los movimientos son hacia la izquierda, entonces los "movimientos" son positivos. Si los movimientos son hacia la derecha, entonces los "movimientos" son negativos. En nuestro ejemplo, el exponente = 128 + 4 = 132 y nos quedamos con .1011011.
3. Cambiar el bit a la derecha de la coma decimal para indicar el signo del número decimal.
   Si el número era positivo entonces el bit se pone a cero.
   Si el número era negativo, entonces el bit se pone a uno.
   En nuestro ejemplo era positivo y se transforma en .0011011.

4. Encontrar la *mantisa*.
   Descartar la coma decimal y dividir el número binario restante en cuatro bytes, añadiendo ceros si es necesario, para componer 32 bits.
   00110110 00000000 00000000 00000000
   y convertirse entonces cada byte otra vez a su número decimal.
   Y así, se puede ver nuestro ejemplo que
   11.375 = 132 45 0 0 0

Si ahora está completamente confundido, como le he dicho, la Spectrum puede ayudarle. La Spectrum tiene un método útil (útil para nosotros) para almacenar variables simples, inmediatamente después del código de las variables, en su forma de coma flotante de cinco-bytes. De modo que todo lo que necesitamos hacer es escribir un programa corto para ingresar (con ENTER) un número, y después recogerlo (con PEEK) en una variable para encontrar su forma de cinco bytes.

```
   5 GO TO 5000
   10 REM  PROGRAMA PARA IMPRIMIR
CUALQUIER NUMERO EN FORMA DE
COMA FLOTANTE
   20 PRINT AT 3,0; "EL SIGUIENTE PRO-
GRAMA CORTO IMPRIMIRA CUALQUIER
NUMERO, DECIMAL, ENTERO O NEGA-
TIVO EN FORMA DE COMA FLOTANTE.
              A LOS NUMEROS ENTEROS SE
LES SUMA EL DECIMAL 0.00000000001
PARA CONVERTIRLOS A UN NUMERO DE
COMA FLOTANTE PERO NO AFECTA A
SU VALOR.
   25 PRINT " "LOS NUMEROS EN FORMA
COMPRIMIDA TAMBIEN PUEDEN USARSE
CON EL LITERAL DE CALCULADORA 34H
(52d)


   40 RETURN
  100 INPUT "NUMERO:"  TAB 9;n
  200 LET n$="NUMERO:"
  300 LET b$="COMA FLOTANTE :"
  400 LET c$="FORMA COMPRIMIDA:"

  500 LET d$="□"
1003 IF N <>0 THEN GO TO 1010
1005 PRINT N$=; TAB 9; d$; PAPER 6;0
'b$; TAB 9; d$; "0  ";d$;"0  ";d$;"0
  ";d$;"0  ";d$;"0"
1006 PRINT c$;TAB 9;d"0  "d;$;"
176";d$;"0"
1007 GO TO 100
1010 LET N=N+.00000000001
1020 PRINT n$;TAB 9;d$; PAPER 6;N
1025 PRINT b$;
```

```
1030 LET X=PEEK 23627+256*PEEK 23
628
1040 FOR A=1 TO 5
1050 PRINT TAB 5+4*a;d$;PEEK (X+A);
1060 NEXT A
1065 PRINT '
1070 DIM a(5)
1075 LET q=0
1076 PRINT c$;
1080 FOR b=5 TO 1 STEP -1
1085 LET a(b)=PEEK ((PEEK 23627+2)
56*PEEK 23628)+b)
1086 IF b=2 THEN GO TO 1100
1090 IF q=0 AND a(b)=0 THEN NEXT b
1100 LET q=q+1
1110 NEXT b
1120 LET q=q-2
1125 LET x=(a(1)-80)
1126 IF x>=64 THEN GO TO 2000
1130 LET e=x+(q*64)
1140 PRINT TAB 9;d$;e;
1150 FOR b=1 TO q+1
1160 PRINT TAB 9+b*4; d$;a(1+b);
1170 NEXT b
1180 PRINT
1190 GO TO 100
2000 PRINT TAB 9; d$;q*64;TAB 13;d
$;a(1)-80;
2010 FOR b=1 TO q+1
2020 PRINT TAB 13+b*4;d$;a(1+b);
2030 NEXT b
2040 PRINT : GO TO 100
5000 DATA 24,79,5,2, "PARE LA CINTA"
5100 DATA 15,167,7,4, "NUMEROS"
5200 DATA 15,130,2,4,"_______"
5250 DATA 24,15,1,2, "PULSE CUALQUIER
TECLA"
5500 DATA 31,170,1,2,"INSTRUCCIONES"
6000 BORDER 45: PAPER 6: INK 0: CL S
6010 LET n=0
6050 RESTORE 5000
6100 FOR a=1 TO 4
6200 INK a: GO SUB 8000
6300 PAUSE 50: NEXT a
6400 PAUSE 0: CLS
6420 INK 2: GO SUB 8000
6450 INK 0: PAPER 7: GO SUB 10
7000 RESTORE 5250: INK 2: GO SUB
8000
7005 IF INKEY$<>"" THEN GO TO 70 05
7006 IF INKEY$="" THEN GO TO 700 6
7010 CLS : GO TO 100
8000 READ x,y,h,w,a$
8010 LET c=USR 32393
8020 RETURN
9998 CLEAR 32334: LOAD "udgs" CODE
 USR "a",168: LOAD ""CODE : GO TO
 5000
9999 SAVEW "NUMBER" LINE 998: SAVE
"udgs" CODE USR "a",168: SAVE "LARGE"
CODE 32335,265
<336   S.A.Nicholls
```

Programa “Números”

Este programa está disponible en cinta y tiene colores, como parte de las series n$ b$ c$ y d$.

Después de RUN, el programa imprimirá los cinco bytes de coma flotante, *y* la forma comprimida de cualquier número. La forma comprimida se discutirá más tarde. Observe que el 0 es tratado como un caso especial en el cual todos los bytes son 0, y también que 0,5 y $^1/_2$ dan resultados distintos.

Hasta ahora se preguntará qué tiene que ver esto con la música y sonido. La respuesta es que la orden BEEP del BASIC hace uso de la pila de la calculadora, y cualquier número almacenado en la pila debe ponerse en formato de cinco-bytes.

Ahora podemos buscar la forma de convertir en código máquina la instrucción BEEP duración, tono. Este método consiste en apilar el valor de la duración y del tono, en formato de cinco-bytes, en la parte superior de la pila de la calculadora y llamar después a la rutina BEEP desde la dirección 03F8h (1016 d).

Si tomamos, por ejemplo, BEEP.1,11, entonces, utilizando el programa NUMBERS podemos convertir .1 y 11 en su forma de cinco-bytes:

.1 = 125 76 204 204 204
11 = 132 48 0 0 0

Ahora debemos encontrar la manera de apilar esos diez bytes consecutivamente en la pila de la calculadora. Un método podría ser encontrar la dirección de comienzo desde la cual los DATA deben ser transferidos mediante PEEK a la variable STACKEND del sistema, y utilizar después una instrucción LDIR tal y como se muestra en el programa 7.1

```
org 2376Ø
CAPITULO 7
PROGRAMA 1

BEEP .1,11

ENCONTRAR STACKEND

2376Ø ED 5B 65 5C      ld de,(23653)
23764 21 E4 5C         ld hl,DATA
23767 Ø1 ØA ØØ         ld bc,1Ø
2377Ø ED BØ            ldir

ACTUALIZAR STACKEND

23772 ED 53 65 5C      ld (23653),de

LLAMADA BEEP

23776 CD F8 Ø3         call 1Ø16
23779 C9               ret
DATA
Ø.1 (forma de 5 bytes)
defb 125 76 2Ø4 2Ø4 2Ø4

11 (forma de 5 bytes)
defb 132 48 Ø Ø Ø
```

Programa 7.1

El RESET es necesario para volver a cargar STACKEND con la nueva dirección de la parte superior de la pila. Recuerde

que LDIR termina con HL y DE conteniendo la *dirección* del último byte transferido + 1. Cuando se llama a la rutina BEEP desde la dirección 03F8 ésta toma los dos valores superiores de la pila y opera con ellos para obtener la duración y el tono requeridos. Se reinicializa la pila para volver al BASIC.

Otro método para apilar números enteros es utilizar LDA,n y la subrutina CALL STACK A de la dirección 2D28h (11560 d) para números entre 0 y 255, ó LDBC,nn y la subrutina CALL STACK BC de la dirección 2D2Bh (115 63 d) para números entre 0 y 65535. Entonces el programa 7.1 puede escribirse como el programa 7.2.

```
org 2376Ø
CAPITULO 7
PROGRAMA 2

BEEP .1,11

ENCONTRAR STACKEND

2376Ø ED 5B 65 5C      ld de,(23653)
23764 21 E9 5C         ld hl,DATA
23767 Ø1 Ø5 ØØ         ld bc,5
2377Ø ED BØ            ldir

ACTUALIZAR STACKEND

23772 ED 53 65 5C      ld (23653),de
23776 3E ØB            ld a,11

APILAR valor a

23778 CD 28 2D         call 1156Ø

CALL BEEP

23781 CD F8 Ø3         call 1Ø16
23784 C9               ret
DATA
Ø.1 (forma de 5 bytes)
defb 125 76 2Ø4 2Ø4 2Ø4
```

Programa 7.2

Por último, para aquellos que deseen algo más que los números de coma flotante, podemos convertirlos en su *forma comprimida* y utilizar el literal 34h(52 d) de la calculadora para apilar datos en esta forma comprimida.

Las etapas para convertir los números de cinco-bytes son las siguientes:

1. Empezar con el quinto byte, sacar todos los números cero hasta que encontremos un número distinto de cero *o* hasta que lleguemos al segundo byte.
2. El cociente es entonces el número de los bytes restantes *menos 1*.
3. Restar 50h (80d) del exponente y, si el resultado es mayor o igual a 40h (64d) entonces ir a la instrucción 6.
4. Cambiar el exponente a (exponente − 80d) + (cociente * 64d).
5. Ir a la instrucción 7.
6. El exponente tiene ahora dos bytes. El primer byte es el cociente * 64d; el segundo byte es el exponente − 80d.
7. Colocar los bytes que quedan, detrás de los bytes del exponente modificados.

Para volver a convertir un número comprimido en su forma de cinco-bytes son necesarias las siguientes etapas:

1. Dividir el primer byte por 64d.
2. Si queda un resto, entonces el exponente será 80d + el resto.
3. Si no hay resto entonces el exponente será 80d + el segundo byte.
4. El cociente es utilizado para encontrar cuantos bytes extra siguen al exponente.

El número de bytes extra es el cociente + 1.

Por ejemplo 11 = 5248 en su forma comprimida:
52 dividido por 64 = 0 con 52
de resto; cociente = 0
bytes posteriores = 1.
Como hubo resto, entonces el exponente = 52 + 80 = 132
1 byte posterior especificado = 48
Rellenar los 5 bytes con 0 0 0
Por lo tanto, 52 48 = 132 48 0 0 0

De nuevo, si todo esto es excesivo para usted por ahora, entonces el programa NUMBERS hará todas las conversiones por usted. Utilizando números en forma comprimida, BEEP .1,11 puede convertirse al código máquina, como se muestra en el programa 7.3.

```
org 23760
CAPITULO 7
PROGRAMA 3


BEEP .1,11

UTILIZAR CALCULADORA

23760 EF                  rst 40

APILAR 0,1 (FORMA COMP.)

defb 52 237 76 204 204 204

STACK 11 (FORMA COMP.)

defb 52 52 48

FIN CALC


defb 56

LLAMADA BEEP

23771 CD F8 03            call 1016
23774 C9                  ret
```

Programa 7.3

En el programa 7.3 RST 28h (40d) llama a la rutina CALCULATOR, y defb 38h (56d) es el literal END CALC.

El FANFARE BEEP para el programa "CROSS" es una combinación de los tres método, a saber, almacenamiento de los números como DATA, utilización de STACK A y utilización del literal 34h (52d) para apilar números en forma comprimida. En este programa se tiene la suerte de que los bytes de la mantisa de los decimales .1, .8 y 0.5 en forma comprimida son iguales,

por lo que sólo se necesita almacenar en DATA el byte del exponente.

```
  1 REM CAPITULO 7
  2 REM PROGRAMA 4
  REM FANFARE BEEP
 1Ø FOR a=1 TO 8
 2Ø READ b,c
 3Ø BEEP b,c
 5Ø NEXT a
 6Ø DATA .1,11,.1,11,.8,16,.Ø5,
11,.Ø5,16,.Ø5,11,.Ø5,16,1,2Ø

org 2376Ø
2376Ø Ø6 Ø7         ld b,7
23762 21 ØØ 5D      ld hl,DATA
LAZO
23765 7E            ld a,(hl)
23766 32 DE 5C      ld (VALOR),a
23769 23            inc hl
2377Ø C5            push bc
23771 E5            push hl
23772 EF            rst 4Ø
defb 52
VALOR
defb Ø 76 2Ø4 2Ø4 2Ø4 56
2378Ø E1            pop hl
23781 7E            ld a,(hl)
23782 E5            push hl
23783 CD 28 2D      call 1156Ø
23786 CD F8 Ø3      call 1Ø16
23789 E1            pop hl
2379Ø C1            pop bc
23791 23            inc hl
23792 1Ø E3         djnz LAZO
23794 3E Ø1         ld a,1
23796 CD 28 2D      call 1156Ø
23799 3E 14         ld a,2Ø
238Ø1 CD 28 2D      call 1156Ø
238Ø4 CD F8 Ø3      call 1Ø16
238Ø7 C9            ret
DATA
defb 237 11 237 11 24Ø 16 236
defb 11 236 16 236 11 236 16
```

Programa 7.4

Ya he mostrado como convertir cualquier orden BEEP del BASIC al código máquina. Esto está bien para hacer música, pero de esta forma no obtendremos efectos de sonido decentes. Para este tipo de sonidos necesitaremos entrar en la rutina BEEP en un punto diferente, 03B5h (949d). Después, los valores de la duración y del tono que han sido sacados de la pila son manipulados, y se termina con el "tono" en el registro HL y la "duración" en el registro DE. De esta forma se podría escribir un programa para conseguir un tono con cadencia incrementando el valor almacenado en HL y llamando a la rutina BEEP repetidamente mediante un lazo "FOR/NEXT". Podemos realizar esto como se muestra en el programa 7.5.

```
org 2376Ø
CAPITULO 7
PROGRAMA 5

SONIDO 1

2376Ø Ø6 FF         ld b,255
23762 21 Ø1 ØØ      ld hl,1
LAZO
23765 11 Ø1 ØØ      ld de,1
23768 E5            push hl
23769 C5            push bc
LLAMADA BEEP

2377Ø CD B5 Ø3      call 949
23773 C1            pop bc
23774 E1            pop hl
23775 23            inc hl
23776 1Ø F3         djnz LAZO

23778 C9            ret
```

Programa 7.5

Cuando ejecute el Programa 7.5 observará que según varíe el tono así cambiará gradualmente la duración. Esto es así debido a que de hecho, la duración es función de los valores de los registros HL y DE. Para igualar la duración debemos decrementar el valor de DE según se incremente el valor de los registros. HL. Así llegamos al Programa 7.6.

```
org 23760
CAPITULO 7
PROGRAMA 6

SONIDO 2

23760 06 64         ld b,100
23762 21 01 00      ld hl,1
23765 11 65 00      ld de,101
LAZO
23768 D5            push de
23769 E5            push hl
23770 C5            push bc
23771 CD B5 03      call 949
23774 C1            pop bc
23775 E1            pop hl
23776 D1            pop de
23777 23            inc hl
23778 1B            dec de
23779 10 F3         djnz LAZO

23781 C9            ret
```

Programa 7.6

Tenemos un último método para producir sonido con la Spectrum, que consiste en enviar "señales" directamente al altavoz conmutándolo a alta o a baja de la misma manera que lo hace la rutina de la ROM. Este método parece poco útil para obtener notas musicales, ya que necesitaríamos conocer la frecuencia de conmutación de alta/baja para cada nota, y de hecho podría ser una repetición de la rutina de la ROM. Sin embargo, se utiliza en programas de juegos para producir un sonido de tipo "ruido blanco" para explosiones; etc.

La instrucción para comunicar con el altavoz es OUT (254),A. Esta instrucción no sólo controla el altavoz, sino también el color de los bordes de la siguiente forma:

1. Si el bit 0,A está a uno, entonces se activa el azul.
   Si el bit 0,A está a cero, entonces se desactiva el azul.
2. Si el bit 1,A está a uno, entonces se activa el rojo.
   Si el bit 1,A está a cero, entonces se desactiva el rojo.
3. Si el bit 2,A está a uno, entonces se activa el verde.
   Si el bit 2,A está a cero, entonces se desactiva el verde.
4. Si el bit 4,A está a uno, entonces el altavoz está en "alta".
   Si el bit 4,A está a cero, entonces el altavoz está en "baja".

Como puede ver, si el bit 4 cambiase de cero a uno a intervalos regulares, entonces se podría producir una nota dependiente de este intervalo.

Para obtener un ruido blanco auténtico necesitaríamos conmutar con un intervalo aleatorio, pero ello requeriría una rutina larga para obtener el número aleatorio y examinar el bit 4,A

cada vez, lo que originaría una pausa "larga" entre cada conmutación. Por lo tanto debemos encontrar un método rápido de obtención de "números aleatorios". Esto no es tan difícil como parece; a nosotros sólo nos interesa el estado del bit 4,A, y para nuestros propósitos los bytes de la ROM son suficientemente aleatorios. Con esto en mente podemos escribir un programa que busque en cada dirección de la ROM de forma sucesiva, y examine el estado del bit 4 del número contenido en cada dirección para conmutar el altavoz a alta o a baja de acuerdo con él.

```
org 23760
23760 21 00 00     ld hl,0
L1
23763 7E           ld a,(hl)
23764 00           nop
23765 D3 FE        out (254),a
2376700            nop
23768 00           nop
23769 D3 FE        out (254),a
23771 00           nop
23772 00           nop
23773 D3 FE        out (254),a
23775 00           nop
23776 23           inc hl
23777 7C           ld a,h
23778 FE 3C        cp 60
23780 20 ED        jr nz,L1
23782 C9           ret
```

Programa 7.7

Hay varios puntos a observar en el programa 7.7.

1. Las tres instrucciones OUT (254),A y NOPs producen un retardo. Usted puede añadirle o quitarle más y escuchar el efecto.
2. La longitud de sonido es gobernada por la instrucción CP 60. Si se redujese a CP 40, por ejemplo, la longitud del sonido se reduciría de forma similar. Cualquier valor inferior a 60 se ejecuta en un área de ROM que no contiene números aleatorios. Pruebe y vea qué sucede.
3. El color de los bordes también queda afectado temporalmente según cambian los bits 0-2 de forma aleatoria. Para que los bordes permanezcan con el mismo color necesita reemplazar los bits 0-2 por los bits 3-5 del valor que contiene la variable BORDER del sistema.

Si quiere experimentar con este método de obtención de música recuerde que se accede 50 veces por segundo a la rutina de interrupción de la ROM y ello alterará su frecuencia, por lo que necesitará deshabilitarla (DI) antes de su rutina, y habilitarla (EI) después. Esto es lo que hace la rutina de la ROM, y explica por qué no se puede producir un BREAK de un programa BASIC durante la ejecución de una orden BEEP.

Dejo el resto a su imaginación e ingenuidad para que altere los valores contenidos en los registro HL y DE al objeto de obtener efectos maravillosos y siniestros, pero para darle una idea de los sonidos que se pueden conseguir usando el método anterior, la cinta contiene el Programa SOUND que tiene cinco so-

nidos diferentes a los que se accede pulsando las teclas de la 1 a la 5. El borde parpadeante produce una dimensión extra de los sonidos 1 y 5, y es muy simple de programar.

Una última cuestión —produzca sonidos tan cortos como sea posible ya que éstos detienen todos los movimientos gráficos durante su ejecución. Los mejores sitios para los sonidos son los lazos de retardo utilizados para relentizar los movimientos gráficos.




```
  10 REM ? RETURN RUN GO SUB X
PRINT THEN G TO ?] LET PRINT THEN O
TO ]]LET PRINT THEN W TO D] LET PRINT
THEN _ TO _] LET PRINT THEN g TO []
LET >? STEP £"? RETURN @ GO SUB X
THEN G>=?<>
2 OR !d?
  25>INK 2: PAPER 7:BORDER 4:CLS
  30 LET x=16: LET y=100: LET w=2: LET
h=4: LET a$="PARE LA CINTA"
  40 PAPER 8: RANDOMIZE USR 32393
  50 PRINT #0;AT 0,3;"PULSE CUAL-
QUIER TECLA PARA EMPEZAR"
  60 PAUSE 0
  65 CLS
  70 LET x=8: LET y=175: LET w=5: LET
h=15: LET a$="SONIDO!"
  80 INK 2: RANDOMIZE USR 32393
  90 PRINT AT 19,2;"Pulse teclas 1 a 5
para sonidos"
```

```
 100 PRINT AT 21,3;"Con SPACE volverá
al BASIC"
 110 RANDOMIZE USR 23760
 120 CLS
 130 LET x=24: LET y=100: LET h=5: LET
w=1: LET a$="To re RUN enter GOTO 65
 140 RANDOMIZE USR 32393
 150 STOP
9998 CLEAR 32334: LOAD ""CODE: GO
TO 25
9999 SAVE "SOUNDS" LINE 9998: SAVE
"LARGE"CODE 32335,265: STOP <336 @
S.A.Nicholls
```

```
Org 23760
COMIENZO
23760 01 FE F7          ld bc,63486
23763 ED 78             in a,(c)
23765 F5                push af
23766 CB 47             bit 0,a
23768 CC 07 5D          call z,BEEP1
23771 F1                pop af
23772 F5                push af
23773 CB 4F             bit 1,a
23775 CC 29 5D          call z,BEEP2
23778 F1                pop af
23779 F5                push af
23780 CB 57             bit 2,a
23782 CC 44 5D          call z,BEEP3
23785 F1                pop af
23786 F5                push af
23787 CB 5F             bit 3,a
23789 CC 5F 5D          call z,BEEP4
23792 F1                pop af

23793 F5                push af
23794 CB 67             bit 4,a
23796 CC 7B 5D          call z,REEP5
23799 F1                pop af
23800 3E 07             ld a,7
23802 CD 9B 22          call 8859
23805 01 FE 7F          ld bc,32766
23808 ED 78             in a,(c)
23810 CB 47             bit 0,a
23812 C8                ret z
23813 18 C9             jr COMIENZO
BEEp1
23815 06 32             ld b,50
L1
23817 C5                push bc
23818 21 64 00          ld hl,100
L2
23821 11 0A 00          ld de,10
23824 E5                push hl
23825 7D                ld a,1
23826 E6 07             and 7
23828 CD 9B 22          call 8859
23831 CD B5 03          call 949
23834 E1                pop hl
23835 11 0A 00          ld de,10
23838 AF                xor a
23839 ED 52             sbc hl,de
23841 7C                ld a,h
23842 B5                or 1
```

```
23843 20 E8          jr nz,L2
23845 C1             pop bc
23846 10 E1          djnz L1
23848 C9             ret
BEEP2
23849 21 00 00       ld hl,0
L3
23852 7E             ld a,(hl)
23853 00             nop
23854 D3 FE          out (254),a
23856 00             nop
23857 00             nop
23858 D3 FE          out (254),a


23860 00             nop
23861 00             nop
23862 D3 FE          out (254),a
23864 00             nop
23865 00             nop
23866 D3 FE          out (254),a
23868 00             nop
23869 23             inc hl
23870 7C             ld a,h
23871 FE 3C          cp 60
23873 20 E9          jr nz,L3
23875 C9             ret
BEEP3
23876 06 05          ld b,5
L5
23878 C5             push bc
23879 21 32 00       ld hl,50
L4
23882 11 0A 00       ld de,10
23885 E5             push hl
23886 CD B5 03       call 949
23889 E1             pop hl
23890 11 32 00       ld de,50
23893 19             add hl,de
23894 7C             ld a,h
23895 FE 04          cp 4
23897 20 EF          jr nz,L4
23899 C1             pop bc
23900 10 E8          djnz L5
23902 C9             ret
BEEP4
23903 06 0A          ld b,10
L6
23905 C5             push bc
23906 21 00 04       ld hl,1024
L7
23909 11 01 00       ld de,1
23912 E5             push hl
23913 CD B5 03       call 949
23916 E1             pop hl
23917 AF             xor a
23918 11 10 00       ld de,16
23921 ED 52          sbc hl,de
23923 7C             ld a,h
23924 B5             or 1
23925 20 EE          jr nz,L7
23927 C1             pop bc
23928 10 E7          djnz L6
23930 C9             ret
BEEP5
```

```
23931 21 Ø1 ØØ        ld hl,1
L8
23934 11 Ø5 ØØ        ld de,5
23937 E5              push hl
23938 CD B5 Ø3        call 949
23941 E1              pop hl
23942 23              inc hl
23943 7C              ld a,h
23944 FE Ø2           cp 2
23946 2Ø F2           jr nz,L8
23948 Ø6 FA           ld b,25Ø
L9
2395Ø C5              push bc

23951 78              ld a,b
23952 E6 Ø6           and 6
23954 CD 9B 22        call 8859
23957 21 9Ø Ø1        ld hl,4ØØ
2396Ø 11 Ø1 ØØ        ld de,1
23963 CD B5 Ø3        call 949
23966 C1              pop bc
23967 1Ø ED           djnz L9
23969 C9              ret
```

Programa "Sound"

8 ATTRIBUTE, SCREEN$ Y POINT

Como usted ya sabe, si conoce la programación de juegos en BASIC, llegará un momento en el que se necesitará encontrar un carácter que está en una posición particular de la pantalla. Esto es útil, por ejemplo, si tiene que examinar si un misil ha dado en el blanco, o si ha conseguido alunizar. Hay tres formas de hacer esto en BASIC, a saber:
Con **ATTRIBUTE** que examina INK: PAPER: FLASH: BRIGHT, es decir, el estado de un cuadro de carácter, y regresa con un número entre 0 y 255.
Con **SCREEN$** que examina el carácter en un cuadro particular (pero sólo los caracteres comprendidos entre CODE 32 y 127); para los restantes caracteres devuelve una serie vacía).
CON **POINT** que examina el estado de un pixel de pantalla: 0 = no dibujado, 1 = dibujado.
Los tres métodos anteriores son disponibles en código máquina haciendo llamadas a rutinas ROM.

ATTRIBUTE (línea, columna)

La rutina en código máquina —call 9603— requiere el número de línea en el registro C y el número de columna en el registro B. El valor de ATTRIBUTE después de llamar a la rutina queda almacenado en la parte superior de la pila de la calculadora. El valor de ATTRIBUTE se calcula de la siguiente forma:
Valor = 128 * (número de FLASH) + 64 * (número de BRIGHT) + 8 (color de PAPER) + color de INK
Puede ver en la fórmula anterior que el valor está comprendido entre 0 y 255 y así puede moverse fácilmente desde la pila al registro A utilizando la rutina "CALC VALUE TO A" —call 11733— y probarse con una instrucción CPn.

El programa 8.1 muestra el método anterior utilizado para PRINT AT 0, 20; PAPER 1; INK 7; FLASH 1; CHR$ 144 (que ha sido redefinido como un hombre). El programa examina entonces cada posición de pantalla, mostrada por un signo >,

hasta que encuentra el valor 143 de ATTR; después vuelve al BASIC.

```
org 2376Ø
CAPITULO 8
PROGRAMA 1

EXAMINAR ATRIBUTOS

ESTABLECER UDG

2376Ø ED 5B 7B 5C      ld de,(23675)
23764 21 42 5D         ld hl,DATA3
23767 Ø1 Ø8 ØØ         ld bc,8
2377Ø ED BØ            ldir

BORRAR PANTALLA

23772 3E Ø2            ld a,2
23774 CD Ø1 16         call 5633
23777 CD 6B ØD         call 3435
2378Ø 3E Ø2            ld a,2
23782 CD Ø1 16         cal 5633

IMPRIMIR UDG

23785 11 2E 5D         ld de,DATA 1
23788 Ø1 1Ø ØØ         ld bc,16
23791 CD 3C 2Ø         call 8252

ESTABLECER OVER 1

23794 FD 36 57 Ø3 ld (iy+87),3

REINICIALIZAR COMIENZO EN Ø/Ø

23798 AF               xor a
23799 32 4Ø 5D         ld (COL),a

IMPRIMIR >

L1
238Ø2 11 3E 5D         ld de, DATA2
238Ø5 Ø1 Ø4 ØØ         ld bc,4

238Ø8 CD 3C 2Ø         call 8252

RETARDO

23811 21 FF FF         1d hl,65535
L2
23814 2B               dec hl
23815 7C               ld a,h
23816 B5               or 1
23817 2Ø FB            jr nz,L2

SACAR >

23819 11 3E 5D         ld de,DATA2
23822 Ø1 Ø4 ØØ         ld bc,4
23825 CD 3C 2Ø         call 8252

INC. COLUMNA

23828 3A 4Ø 5D         ld a,(COL)
23831 3C               inc a
23832 32 4Ø 5D         ld (COL),a

OBTENER COLUMNA/LINEA en BC

23835 ED 4B 3F 5D ld bc, (LINEA)

LLAMADA ATTR. &
OBTENER VALOR EN A

23839 CD 83 25         call 96Ø3
23842 CD D5 2D         call 11733

EXAMINAR SI 143

23845 FE 8F            cp 143

MOVERSE A SIGUIENTE POS.

23847 2Ø D1            jr nz,L1

REINICIALIZAR OVER Ø

23849 FD 36 57 ØØ ld (iy+87),Ø
23853 C9               ret

IMPRIMIR DATOS UDG

DATA1

defb 22 Ø 2Ø 17 1 16 7 18 1
defb 144 18 Ø 17 7 16 Ø

PRINT > DATA

DATA2
defb 22
LINEA
defb Ø
COL
defb Ø 62

DATOS UDG

DATA3
defb 24 153 126 153 24 36 36 1Ø2
```

Programa 8.1

Cuando se ejecuta, el programa 8.1 da la apariencia de un misil moviéndose lentamente hacia un hombre, y vuelve al BASIC cuando choca con el hombre. Por supuesto, podría utilizar una rutina de choque en vez de la instrucción RETURN (con efectos de sonido). Esta forma de examinar las posiciones de pantalla para blancos, etc., está bien, pero es algo limitada porque todos los blancos deben tener el mismo valor de ATTRIBUTE.

SCREEN$ (línea, columna)

Al igual que con la rutina ATTR el número de línea se guarda en el registro C, y el número de columna en el registro B antes de hacer una llamada a 9528. La rutina guarda los *parámetros de la serie* en la parte superior de la pila de la calculadora en forma de cinco-bytes. El uso de la pila de la calculadora para guardar los parámetros de la serie es una buena idea puesto que hasta ahora sólo la hemos usado para guardar números en forma de cinco-bytes. Hay que decir que hay literales de calculadora para manipular series, pero en vez de poner la serie en la pila, sus "parámetros" se guardan en forma de cinco-bytes. Para poner los parámetros de la serie en la pila, el par de registros BC guarda la longitud de la serie, y el par de registros DE guarda la dirección de comienzo de la serie. Para series sencillas, el registro A almacena el valor cero. Una llamada a la rutina —call 10929— colocará los parámetros en la pila en el orden correcto. De forma similar, la rutina —call 11249— tomará el último "valor" de la pila y lo colocará en los registros A, B, C, D y E; de nuevo BC contendrá la longitud de la serie, y DE la dirección de comienzo.

El programa 8.2 muestra las rutinas que se usan para imprimir el carácter @ en la posición SCREEN 0,20; y cuando se ejecuta termina con dos @ en 0,20; y 0,21. En vez de PRINT STRING —call 8252— podríamos haber utilizado LD A,(DE) para examinar si el carácter anual está en la posición 0, 20; y después realizar las acciones oportunas.

Sin embargo, hay una limitación en la rutina SCREEN$ del BASIC, y es que sólo encontrará caracteres comprendidos entre CODE 32 y CODE 127; para cualquier otro valor se obtendrá una serie vacía. Esto excluye la utilización de este método para encontrar UDGs, y en BASIC esto es una gran contrariedad para los programas de juegos. En código máquina esto no es problema, puesto que hay formas "rápidas" de engañar a la ROM y de colocar los UDGs en el conjunto de caracteres. La

variable del sistema CHARS-23606/7 normalmente guarda la dirección de comienzo del conjunto de caracteres −256, es decir 15360, y es utilizada por la rutina SCREEN$ para encontrar el carácter. Podemos establecer esta variable para que apunte al comienzo de los UDGs −256 antes de que se haga la llamada a SCREEN$ (convirtiendo UDG"A" en CODE 32, UDG"B" en CODE 33, etc.) y después reinicializarla a su valor normal. De esta forma podemos examinar los UDGs para encontrar el carácter que queremos.

```
org 23760
CAPITULO 8
PROGRAMA 2

SCREEN$ (0,20)

BORRAR PANTALLA

23760 3E 02          ld a,2

23762 CD 01 16       call 5633
23765 CD 6B 0D       call 3435
23768 3E 02          ld a,2
23779 CD 01 16       call 5633

IMPRIMIR EN 0,20 @

23773 3E 16          ld a,22
23775 D7             rst 16
23776 3E 00          ld a,0
23778 D7             rst 16
23779 3E 14          ld a,20
23781 D7             rst 16
23782 3E40           ld a,64
23784 D7             rst 16

LLAMADA SCREEN$

23785 06 14          ld b,20
23787 0E 00          ld c,0

23789 CD 38 25       call 9528

LLAMADA 'VALOR A A, B, C, D, E

23792 CD F1 2B       call 11249

LLAMADA IMPRIMIR SERIE

23795 CD 3C 20       call 8252
23798 C9             ret
```

Programa 8.2

El programa 8.3 es lo mismo que el programa 8.1 pero esta vez estamos buscando CHR$ 144 antes de volver al BASIC.

```
org 23836
CAPITULO 8
PROGRAMA 3

SCREEN$ PARA UDG's

23760 TO 23835
IGUAL QUE PROG. 1

ESTABLECER CARACTS. A UDG's -256

23836 2A 7B 5C       ld hl,(23675)
23839 25             dec h
23840 22 36 5C       ld (23606),hl

EXAMINAR SCREEN$

23843 CD 38 25       call 9528
23846 CD F1 2B       call 11249

EXAMINAR SI ES CHR$ (32)
por ejemplo CHR$ 144 (UDG 'A')

23849 1A             ld a,(de)
23850 FE 20          cp 32

REINICIALIZAR CARACTS. A NORMAL

23852 21 00 3C       ld hl,15360
23855 22 36 5C       ld (23606),hl

RESTO DEL PROGRAMA
IGUAL QUE PROGRAMA 1

23847 PARA TERMINAR DATA3
```

Programa 8.3

Por supuesto, podría copiar el conjunto de caracteres en RAM y mantener a CHARS apuntando permanentemente al comienzo −256. De esta forma puede olvidarse de los UDGs y redefinir cualquier carácter que desee en la nueva serie (deje sin cambiar las letras y los números para que el texto no se altere). Este método conseguirá alrededor de 50 caracteres redefinibles si se limita a utilizar el texto con *letras mayúsculas* y un mínimo de signos de puntuación. La rutina SCREEN$ reconocerá estos caracteres cuando aparezcan en la nueva serie de caracteres.

Hay un punto muy importante a considerar cuando utilice la calculadora para manejar series, y es que se hace uso del espacio de trabajo para guardar la serie y/o el resultado del cálculo. Si este espacio de trabajo no se borra regularmente entonces puede encontrarse con que en su programa aparece un mensaje de fuera de memoria sin razón aparente, o peor todavía, el espacio de trabajo se ejecutará en su rutina de código máquina y su programa se descontrolará. En la práctica es aconsejable borrar el espacio de trabajo cada vez que se llama a la rutina SCREEN$. La rutina de la ROM —call 5823— borra para usted el espacio de trabajo (y también borra la pila de la calculadora). Así, el programa 8.3 debería llamar a 5823 después de LD A,(DE) pero recuerde que debe guardar el valor de A antes de hacer la llamada.

POINT

La instrucción POINT x,y, del BASIC vuelve con un valor de 1 si el pixel x, y está establecido (es decir, dibujado) o 0 si no lo está (PLOT OVER). La rutina en código máquina —call 8910— requiere que la coordenada x se guarde en el registro C y la coordenada y en el registro B antes de hacer la llamada. Al igual que con las rutinas anteriores el resultado termina con el valor en la pila de la calculadora, y, como este valor es 0 ó 1 podemos sacarlo de la pila y ponerlo en el registro A para compararlo con un valor requerido. El programa 8.4 muestra como se utiliza esto para examinar la posición de pantalla 0,175, es decir, la esquina superior izquierda, primero imprime un cuadro negro en 0,0; después vuelve al BASIC como POINT 0,175 = 1. Esta rutina se utiliza en el programa "balloon" para apuntar (con POINT) a cada una de las posiciones del balón antes de redibujar el balón. Si la rutina vuelve con un valor de 1 entonces se llama a la rutina HIT.

```
org 23760
CAPITULO 8
PROGRAMA 4
POINT x,y

23760 3E 02          ld a,2
23762 CD 01 16       call 5633
23765 CD 6B 0D       call 3435
23768 3E 02          ld a,2
23770 CD 01 16       call 5633
23773 3E 8F          ld a,143
23775 D7             rst 16
23776 0E 00          ld c,0
23778 06 AF          ld b,175
23780 CD CE 22       call 8910
23783 CD D5 2D       call 11733
23786 FE 01          cp 1
23788 C8             ret z
```

Programa 8.4

9 LA IMPRESORA

No hay muchos programas que hagan uso de la impresora, pero puede ser conveniente tener algunas veces una copia de la pantalla (para demostrar a los amigos que usted ha logrado las puntuaciones máximas, etc.). Con esto en mente trataremos tres órdenes del BASIC: COPY, LPRINT, y LLIST y sus equivalentes en código máquina.

COPY

Normalmente la rutina COPY pasará directamente los contenidos de las primeras 22 líneas de la pantalla a la impresora. Sin embargo, en código máquina podemos mejorar esto y especificar cuantas líneas de pantalla queremos, y la línea de comienzo; así por ejemplo, podríamos copiar sólo tres líneas empezando a partir de la línea siete.

El programa 9.1 muestra el método utilizado para copiar las 24 líneas, ya que algunos programas de juegos tienen la puntuación en las dos líneas inferiores de la pantalla y no se imprimirán con una orden directa de COPY. Puede ver que antes de llamar a 3762, el par de registros HL contienen la línea de dirección de comienzo, en nuestro ejemplo —esta es la línea 1, y el registro B guarda el número de líneas requerido × 8 (24 × 8 = 192).

```
org 2376Ø                              EJEMPLO 24 LINEAS
CAPITULO 9
PROGRAMA 1                             2376Ø 21 ØØ 4Ø      ld hl,16384
                                       23763 Ø6 CØ         ld b,192
COPIAR n LINEAS DE PANTALLA            23765 F3            di
                                       23766 CD B2 ØE      call 3762
HL=LINEA COMIENZO PANTALLA             23769 C9            ret
  B=LINEAS ×8
```

Programa 9.1

LPRINT

Esta instrucción puede considerarse como la misma que PRINT con la única diferencia de que los caracteres de control no fun-

cionarán, por supuesto, y de que requiere que se abra un canal diferente antes de la instrucción RST 16 para enviar la salida a la impresora en vez de a la pantalla. Así, para LPRINT "A", por ejemplo, primero hay que emplear OPEN CHANNEL 3, LD A, CODE "A" y después RST 16 para que se imprima.

El programa 9.2 muestra el método anterior utilizado para imprimir (con LPRINT) una serie de caracteres. Si desea tabular su impresión con LPRINT, el número de TAB debe ponerse, con POKE, en la variable PRCC 23680 del sistema antes de la instrucción RST 16.

```
org 23760
CAPITULO 9
PROGRAMA 2

LPRINT

ABRIR CANAL 3

23760 3E 03          ld a,3
23762 CD 01 16       call 5633
23765 11 DF 5C       ld de,DATA
23768 01 11 00       ld bc,17

LPRINT SERIES

23771 CD 3C 20       call 8252
23774 C9             ret
DATA
defs Este es el programa 2
```

Programa 9.2

LLIST

Aunque no encuentro ninguna aplicación en la que se requiera un listado a la mitad de un programa, he incluido éste porque es una de las órdenes de impresora y es fácil de programar. Todo lo que se necesita es llamar a 6133. El programa 9.3 muestra esto ensamblando el programa en BASIC y después listándolo con LLIST.

```
org 23760
CAPITULO 9
PROGRAMA 3

LLIST

23760 CD F5 17       call 6133
23763 C9             ret

  10 IMPRIMIR PASOS
  20 REM go
  30>REM org 23760
  40 REM ! CAPITULO 9; !PROGRAMA 3;!
  50 REM ! LLIST ;!
  60 REM call 6133
  70 REM ret
9000 REM finish
9010 RANDOMIZE USR 58000
9020 COPY
9030 LPRINT
9040 RANDOMIZE URS 23760
```

Programa 9.3

10 CONVERSION DE PROGRAMA

Si ha logrado llegar a este Capítulo comprendiendo todos los anteriores entonces le felicito —éste no le va a ocasionar ningún problema—. Si no está demasiado seguro acerca de cualquiera de las rutinas anteriores, le aconsejo que vuelva a leer el Capítulo en cuestión antes de empezar éste.

En este capítulo espero mostrar la forma de hacer un programa de juegos enteramente en código máquina utilizando un programa en BASIC como guía.

Sin duda usted tendrá su propia manera de hacer un programa en BASIC. Tal vez tiene una idea y la desarrolla en la Spectrum añadiendo rutinas conforme las va pensando, y termina con un programa completamente desestructurado, pero que funciona, y que sólo usted puede seguir. Este método es satisfactorio, pero puede darse cuenta de que si intenta arreglar el programa unas semanas después ni siquiera *usted* podrá seguirlo. Creo que el mejor método es hacer un organigrama de la idea básica y después añadir las rutinas que sean necesarias, acabando con un diagrama de flujo completo y estructurado a partir del cual puede escribirse el programa en BASIC, y tal vez mejorarse con color, sonido y UDGs. Y lo más importante, así tendrá un programa fácil de convertirse al código máquina.

El programa que convertiremos en este Capítulo se llama "CROSS" y empieza como programa para controlar a un hombre que pasa obstáculos hacia un lugar seguro. Con esta idea se hizo el organigrama. Supone que la rutina principal establece las variables, imprime las instrucciones y efectúa la visualización, y entonces un lazo examina el movimiento del hombre y su posición "nueva" de pantalla. Hay dos puntos de salida de este lazo principal:

1. Si el hombre es alcanzado ingresamos la rutina HIT y examinamos las vidas que le quedan; si no le quedan vidas el juego termina, en caso contrario volvemos a entrar en el lazo principal.
2. La rutina HOME, que también examina el número de lugares a salvo llenados, y añade arañas, o incrementa la velocidad antes de volver al lazo principal.

A partir de este diagrama de flujo se hizo la versión de

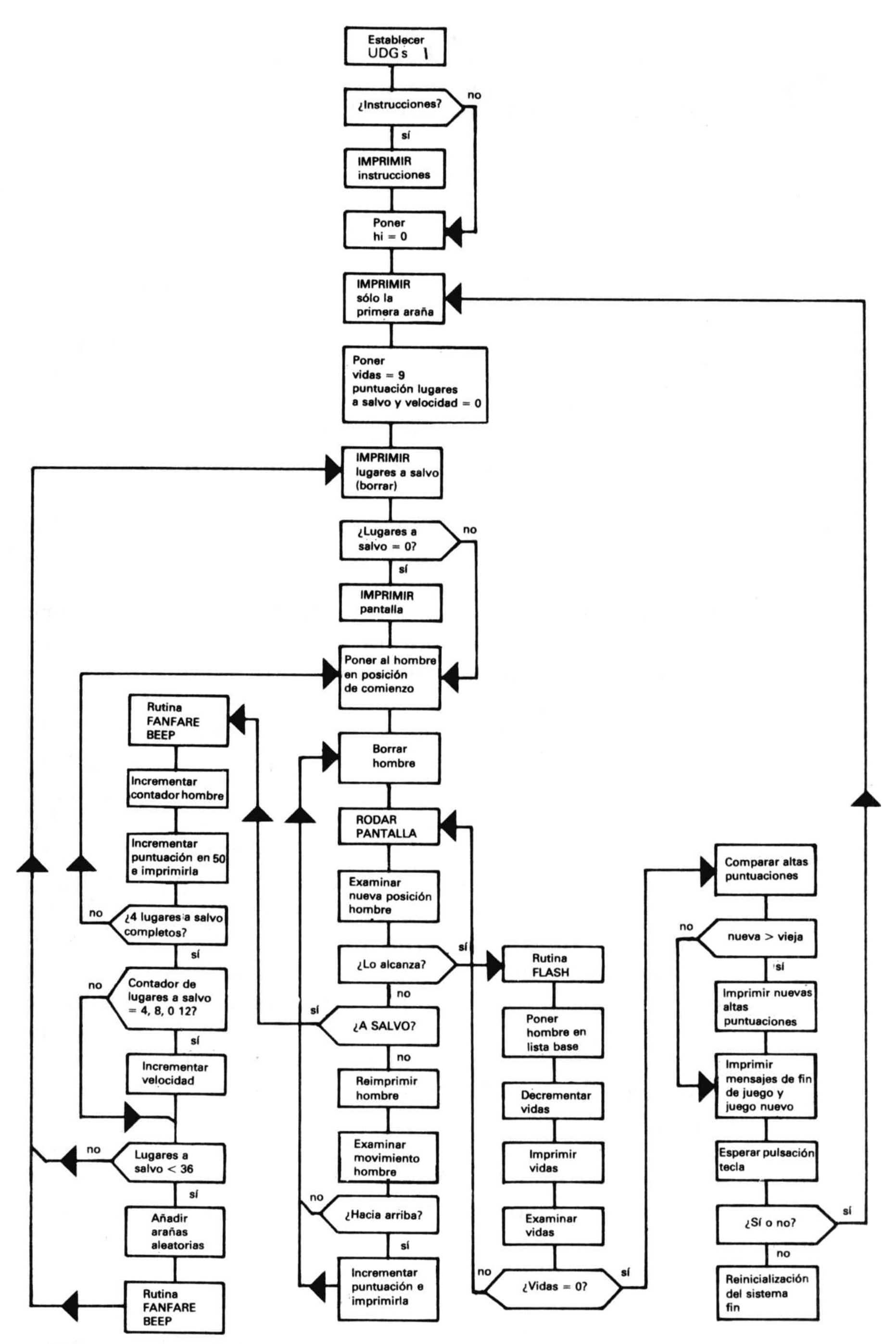


Figura 10.1 Diagrama de Flujo para el programa "Cross".

"CROSS" en BASIC. Si compara el programa actual con el diagrama de flujo podría ser capaz de reconocer todas las rutinas.

El programa, aunque es muy sencillo, se mejora mucho utilizando los UDGs, colores, y sonido. Hace un poco de trampa utilizando una rutina en código máquina para hacer rodar líneas de la pantalla a la izquierda o a la derecha un pixel cada vez, pues una versión en BASIC de esto aumentaría su lentitud de forma increíble.

Se ingresa el código máquina con POKE en la dirección de memoria correcta mediante el programa en BASIC. Esto se podría haber hecho con una instrucción LOAD código "máquina" XX, n para cargar el código directamente desde la cinta.

Antes de continuar con este Capítulo asegúrese de que comprende completamente el funcionamiento del programa, ya que ahora lo analizaremos línea por línea para convertirlo al código máquina.

```
  20 CLEAR 32243: GO TO 40
  25 BEEP .01, b-a
  30 PRINT OVER 1; PAPER 8; INK 8; AT a, y2;
"Ŕ": RETURN
  40 PRINT AT 11, 5;"por favor espere un mo-
mento"
  50 FOR a=32244 TO 32494
  60 READ b: POKE a, b: NEXT a
  70 DATA 14, 8, 229, 17, 31, 0, 25, 126, 237,
82, 31, 6, 32, 126, 31, 119, 35, 16, 250, 225, 36,
13, 32, 234, 201
  80 DATA 14, 8, 175, 229, 17, 31, 0, 237, 82,
126, 25, 23, 6, 32, 126, 23, 119, 43, 16, 250,
225, 36, 13, 32, 233, 201
  90 DATA 33, 95, 64, 205, 13, 126, 33, 128, 64,
205, 244, 125, 33, 128, 64, 205, 244, 125, 33,
223, 64, 205, 13, 126, 33, 0, 72, 205, 244, 125,
33, 0, 72, 205, 244, 125, 33, 0, 72, 205, 244, 125
 100 DATA 58, 121, 92, 0, 0, 0, 0, 0, 0, 0, 230,
2, 40, 20, 33, 64, 72, 205, 244, 125, 33, 64, 72,
205, 244, 125, 33, 64, 72, 205, 244, 125, 24, 18,
33, 95, 72, 205, 13, 126, 33, 95, 72, 205, 13,
126, 33, 95, 72, 205, 13, 126
 110 DATA 33, 128, 72, 205, 244, 125, 33, 192,
72, 205, 244, 125, 33, 192, 72, 205, 244, 125,
33, 31, 80, 205, 13, 126, 33, 31, 80, 205, 13,
126, 33, 95, 80, 205, 13, 126, 201
 120 DATA 33, 128, 72, 205, 244, 125, 33, 192,
72, 205, 244, 125, 33, 31, 80, 205, 13, 126, 33,
95, 80, 205, 13, 126, 201
 130 DATA 33, 95, 64, 205, 13, 126, 33, 128, 64,
205, 244, 125, 33, 0, 72, 205, 244, 125, 201
 140 DATA 33, 95, 64, 205, 13, 126, 33, 223, 64,
205, 13, 126, 33, 128, 72, 205, 244, 125, 33, 192,
72, 205, 244, 125, 201
 150 LET a=PEEK 23675+256*PEEK 23676
 160 FOR b=a TO a+167
 170 READ c: POKE b, c: NEXT b
 180 DATA 15, 18, 34, 127, 255, 40, 16, 128, 64,
32, 254, 254, 255, 40, 16
 190 DATA 127, 127, 127, 127, 127, 255, 21, 8,
254, 254, 254, 254, 255, 255, 64, 128
 200 DATA 0, 248, 196, 196, 254, 254, 40, 16,
24, 24, 36, 126, 60, 90, 165, 66
 210 DATA 56, 40, 146, 124, 56, 56, 40, 108, 1,
2, 4, 127, 127, 255, 20, 8
 220 DATA 240, 72, 68, 254, 255, 255, 20, 8, 0,
31, 35, 35, 127, 127, 20, 8
 230 DATA 127, 127, 127, 127, 255, 255, 2, 1,
254, 254, 254, 254, 254, 255, 168, 16
```

```
240 DATA 16, 41, 199, Ø, 38, Ø, Ø, Ø, Ø, 68,
255, 68, 68, 255, 68, Ø
250 DATA Ø, 34, 85, 143, 151, 163, 16
Ø, Ø, Ø, 68, 17Ø, 241, 233, 197, 5, Ø
260 DATA 16, 16, 16, 254, 63, 31, 15,
7, Ø, Ø, Ø, Ø, 3Ø, 255, 255, 255
27Ø DATA 96, 124, 84, 12Ø, 127, 255, 254, 252,
Ø, Ø, 3, 2, 15, 63, 255, Ø, 6, 12, 152, 24Ø, 224,
85, 255, Ø
28Ø PRINT AT 11, 3;"¿Desea instrucciones?";
AT 13, 11;" si (y) es"; AT 15, 11;"(n)o"
29Ø PAUSE Ø: IF INKEY$="y" THEN GO TO
3ØØ
295 GO TO 4ØØ
3ØØ CLS : PRINT AT Ø, 11;"OBJETIVO" ''"Guiar
un   alrededor de una calle y un río evitando
                 .'''"Una     vigila la isla
central."
31Ø PRINT '"Hay 4 lugares a salvo que pue-
den usarse, por ejemplo, huecos en parte
superior barreras ####  ####"
32Ø PRINT '"Una vez que los 4 lugares a salvo
se han ocupado se incrementará la velocidad,
se añadirá una    extra y los LUGARES A
SALVO estarán vacios."
33Ø PRINT '''AT 18, 9; "Pulse cualquier tecla.:
PAUSE Ø
37Ø CLS : PRINT AT 7, 11; "CONTROLES"''"
375 PRINT ´" ↑
   < >"
38Ø PRINT FLASH 1;AT 11, 6;"1"; FLASH Ø; " 2
3 4 5 6 7 8 "; FLASH 1;"Ø"
39Ø PRINT AT 18, 5; "Press any key to PLAY":
PAUSE Ø
4ØØ BRIGHT 1: PAPER 5: BORDER 5: CLS
41Ø LET hi=Ø
42Ø PRINT PAPER 4;AT 1Ø, Ø;"
                                 "
43Ø LET lives=9: LET score=Ø:
LET home=Ø
44Ø POKE 32425, 2Ø1: POKE 3245Ø, 2Ø1:
POKE 32469, 2Ø1
45Ø PRINT AT Ø, Ø; PAPER 4; "######
"; PAPER 7;" "; PAPER 4;" ##########"
; PAPER 7;" "; PAPER 4;" ##########"
; PAPER 7; " "; PAPER 4;"##########"
; PAPER 7;" "; PAPER 4;"######"
455 IF home <>Ø THEN GO To 66Ø
46Ø PRINT PAPER 4; INK 5;"
                                   "
47Ø PRINT "
                  "
48Ø PRINT INK 7;"
                  "
49Ø PRINT INK 2;"
                  "
5ØØ PRINT INK 7;"
                  "
51Ø PRINT INK 1;"
                  "
52Ø PRINT INK 7;"
                  "
53Ø PRINT "
                  "
54Ø PRINT PAPER 4;"        ####################
################################"
55Ø PRINT PAPER Ø; INK 7;AT 11,Ø;"
    ################################################
#####"
56Ø PRINT PAPER Ø; INK 3;"
                  "
57Ø PRINT PAPER Ø; INK 7;"          -----
------------------------------"
58Ø PRINT PAPER Ø; INK 5;"
                  "
59Ø PRINT PAPER Ø; INK 7;"          =====
=============================="
```

```
 600 PRINT PAPER 0; INK 4;"

 610 PRINT PAPER 0; INK 7;"                      -----
--------------------------------"
 620 PRINT PAPER 0; INK 6;"

 630 PRINT PAPER 4;"          ####################
################################"
 640 PRINT PAPER 4;"

 650 PRINT PAPER 1, INK 7;" SCORE ";AT 21,
11;" HOMBRES "; PAPER 5; INK 0; vidas; PAPER
1; INK 7;" HISCORE "
 660 LET x1=20: LET y1=16: LET x2=x1: LET
y2=y1
 670 PRINT PAPER 8; INK 8; AT x1, y1; ""
 680 RANDOMIZE USR 32295
 690 IF SCREEN$ (x2, y2)=" " THEN GO TO 880
 700 LET a=x2: FOR b=25 TO 35: GO SUB 25:
GO SUB 25: NEXT b
 730 FOR a=x2 TO 20 STEP 2: GO SUB 25: GO
SUB 25: NEXT a
 740 LET lives=lives-1: PRINT AT 21, 16; lives
 750 LET x2=20
 760 IF lives<>0 THEN GO TO 680
 770 IF hi>score THEN GO TO 790
 780 LET hi=score: PRINT AT 21, 27;hi
 790 PRINT FLASH 1; PAPER 7;AT 12, 0;"
"              GAME OVER
 800 PRINT AT 14, 0;" Another game? (y)es (n)o
 840 IF INKEY$="n" THEN RANDOMIZE USR 0
 850 IF INKEY$<>"y" THEN GO TO 840
 860 PRINT PAPER 5;AT 21, 7;"": GO TO 415
 880 IF x2<>0 THEN GO TO 1050
 890 PRINT PAPER 8; INK 8;AT x1, y1;" ": PRINT
AT x2, y2;" "
1020 IF SCREEN$ (10, a+1)="" THEN
 900>RESTORE 920
 910 FOR a=1 TO 8: READ b, c: BEEP b, c:
NEXT a
 920 DATA .1, 11, .1, 11, .8, 16, .05, 11, .05, 16,
.05, 11, .05, 16, 1, 20
 930 LET home=home+1: LET
score=score+50: PRINT AT 21, 7;score
 950 IF home/4<>INT (home/4) THEN GO TO
660
 960 IF home=4 THEN POKE 32425, 0
 970 IF home=8 THEN POKE 32450, 0
 980 IF home=12 THEN POKE 32469, 0
 985 IF home>36 THEN GO TO 450
 990 LET a=RND*30
1000 LET a=a+1
1005 IF a>31 THEN LET a=0
1010 IF SCREEN$ (10, a)="" THEN GO TO1000
1020 IF SCREEN$ (10, a+1) ="" THEN GO TO
1000
1030 PRINT PAPER 4;AT 10, a; "  "
1035 RESTORE 920: FOR a=1 TO 8: READ b, c:
BEEP b,c: NEXT a
1040 GO TO 450
1050 PRINT PAPER 8; INK 8;AT x2,y2;" "
1060 LET x1=x2: LET y1=y2
1070 IF INKEY$<>"1" THEN GO TO 1100
1080 BEEP .001,33
1090 LET x2=x2-2: LET score=score+5: PRINT
AT 21,7;score
1100 LET y2=y2+(INKEY$="0" AND y
2<>31)-(INKEY$="9" AND y2<>0)
1110 GO TO 670
```

GRAFICOS DEFINIBLES POR EL USUARIO :

AB =
CDE =
f =
G =
HI =
JKL =

M =
N =
OP =
QRS =
TU =

Programa "Cross basic"

```
org 32244
CAPITULO 1Ø
PROGRAMA>CROSS>

[ BASIC ] CODIGO/M


RODAR LINEA A DERECHA

DERECHA
32244 ØE Ø8          ld c,8
L1
32246 E5             push hl
32247 11 1F ØØ       ld de,31
3225Ø 19             add hl,de
32251 7E             ld a,(hl)
32252 ED 52          sbc hl,de
32254 1F             rra
32255 Ø6 2Ø          ld b,32
L2


32257 7E             ld a,(hl)
32258 1F             rra
32259 77             ld (hl),a
3226Ø 23             inc hl
32261 1Ø FA          djnz L2
32263 E1             pop hl
32264 24             inc h
32265 ØD             dec c
32266 2Ø FA          jr nz,L1
32268 C9             ret


RODAR LINEA A IZQUIERDA

IZQUIERDA
32269 ØE Ø8          ld c,8
L3
32271 AF             xor a
32272 E5             push hl
32273 11 1E ØØ       ld de,31
32276 ED 52          sbc hl,de
32278 7E             ld a,(hl)

32279 19             add hl,de
3228Ø 17             rla
32281 Ø6 2Ø          ld b,32
L4
32283 7E             ld a,(hl)
32284 17             rla
32285 77             ld (hl),a
32286 2B             dec hl
32287 1Ø FA          djnz L4
32289 E1             pop hl
3229Ø 24             inc h
32291 ØD             dec c
32292 2Ø E9          jr nz,L3
32294 C9             ret

RUTINA DE COMIENZO

HL contiene :-
Línea de comienzo para RODAR a DERECHA
Línea de final para RODAR a IZQUIERDA


VELOCIDAD Ø

32295 21 5F 4Ø       ld hl,16479
32298 CD ØD 7E       call IZQUIERDA
323Ø1 21 8Ø 4Ø       ld hl,16512
323Ø4 CD F4 7D       call DERECHA
323Ø7 21 8Ø 4Ø       ld hl,16512
3231Ø CD F4 7D       call DERECHA
32313 21 DF 4Ø       ld hl,166Ø7
32316 CD ØD 7E       call IZQUIERDA
32319 21 ØØ 38       ld hl,18432
32322 CD F4 7D       call DERECHA
32325 21 ØØ 48       ld hl,18432
32328 CD F4 7D       call DERECHA
32331 21 ØØ 48       ld hl,18432
32334 CD F4 7D       call DERECHA

Mover SPIDER línea
Utiliza contador de TRAMPAS


32337 3A 79 5C       ld a,(23673)
3234Ø ØØ             nop
32341 ØØ             nop
32342 ØØ             nop
32343 ØØ             nop
32344 ØØ             nop
32345 ØØ             nop
32346 ØØ             nop
32347 E6 Ø2          and 2
32349 28 14          jr z,L5
32351 21 4Ø 48       ld hl,18496
32354 CD F4 7D       call DERECHA
32357 21 4Ø 48       ld hl,18496
3236Ø CD F4 7D       call RIGHT
32363 21 4Ø 48       ld hl,18496
32366 CD F4 7D       call DERECHA
32369 18 12          jr L6
L5
32371 21 5F 48       ld hl,18527
32374 CD ØD 7E       call IZQUIERDA
32377 21 5F 48       ld hl,18527
```

```
32380 CD 0D 7E          call IZQUIERDA
32383 21 5F 48          ld hl, 18527
32386 CD 0D 7E          call IZQUIERDA
L6
32389 21 80 48          ld hl, 18560

32392 CD F4 7D          call DERECHA
32395 21 C0 48          ld hl,18624
32398 CD F4 7D          call DERECHA
32401 21 C0 48          ld hl,18624
32404 CD F5 7D          call DERECHA
32407 21 1F 50          ld hl,20511
32410 CD 0D 7E          call IZQUIERDA
32413 21 IF 50          ld hl,20511
32416 CD 0D 7E          call IZQUIERDA
32419 21 5F 50          ld hl,20575
32422 CD 0D 7E          call IZQUIERDA
32425 C9                ret

VELOCIDAD 1

32426 21 80 48          ld hl,18560


32429 CD F4 7D          call DERECHA
32432 21 C0 48          ld hl,18624
32435 CD F4 7D          call DERECHA
32438 21 1F 50          ld hl, 20511
32441 CD 0D 7E          call IZQUIERDA
3244 21 5F 50           ld hl,20575
32447 CD 0D 7E          call IZQUIERDA
32450 C9                ret

VELOCIDAD 2

32451 21 5F 40          ld hl,16479
32454 CD 0D 7E          call IZQUIERDA
32457 21 80 40          ld hl,16512
32460 CD F4 7D          call DERECHA
32463 21 00 48          ld hl,18432
32466 CD F4 7D          call DERECHA
32469 C9                ret

VELOCIDAD 3

32470 21 5F 40          ld hl,16479
32473 CD 0D 7E          call IZQUIERDA
32476 21 DF 40          ld hl,16607
32479 CD 0D 7E          call IZQUIERDA
32482 21 80 48          ld hl,18560
32485 CD F4 7D          call DERECHA
32488 21 C0 48          ld hl,18624
32491 CD F4 7D          call DERECHA
32494 C9                ret
```

Programa "Machine Code for Cross basic"

Conversión en código máquina

LINEA 20 DEL PROGRAMA EN BASIC

No será necesario convertirla a la versión en código máquina final porque no habrá necesidad de reinicializar RAMTOP para proteger el código máquina.

LINEAS 25 Y 30 DEL PROGRAMA EN BASIC

Esta es una subrutina utilizada en la rutina HIT y en código máquina se colocará con la rutina HIT. La rutina se coloca al comienzo del programa en BASIC para reducir el tiempo que tarda la Spectum en encontrarla. Cuando se llama a una subrutina, la Spectrum empieza desde el principio del programa en BASIC y recorre los números de línea necesarios hasta que se localiza la rutina requerida. Puede ver que si tuviese un número de línea alto llevaría más tiempo encontrarla, por lo que se incrementaría el tiempo empleado para ejecutar las líneas desde la 700 hasta la 730, que llaman a esta rutina 22 veces por lo menos.

LINEA 40 DEL PROGRAMA EN BASIC

En BASIC la Spectrum tarda varios segundos en cargar con LOAD el código máquina y establecer los UDGs, por lo que se visualiza este mensaje durante ese período. Sin embargo, en código máquina esto es instantáneo, y el mensaje es por lo tanto innecesario.

LINEAS 50 A 140 DEL PROGRAMA EN BASIC

Esta es la rutina para ingresar, con POKE, el código máquina de la "RUTINA ROLL" en las direcciones de memoria desde 32244 a 32499, y tampoco se necesitará en código máquina, ya que estará localizada correctamente con el resto del programa en código máquina.

LINEAS 150 A 270 DEL PROGRAMA EN BASIC PARA ESTABLECER UDGS

Esta es la primera rutina que necesitaremos convertir. Por supuesto podríamos utilizar SAVE "UDG" CODE USR "a", 168: LOAD "UDG" CODE USR "a" 168, pero ello supondría tener que guardar y cargar dos grupos de rutinas en código máquina, que son el juego principal y los UDGs. Al objeto de que pueda amoldarse fácilmente a la Spectrum de 16K empezaremos el programa en código máquina en la dirección 5DC1 h(24001d). Como en el BASIC, pondremos cada byte de UDGs en forma de DATA y leeremos estos DATA transfiriendo cada número a su lugar adecuado en el área UDG de la memoria. Como verá, el método más simple es utilizar la instrucción LDIR. El par de registros DE se carga con la dirección de comienzo de UDG según lo que contenga la variable 5C7Bh(23675d) del sistema, y el par de registros HL contienen la dirección de comienzo de DATA. La rutina DATA = 5DC1h a 5E68h y la rutina en código máquina = 5E69h(24169 d) a 5E73h(24180d).

Observe que se llama a la rutina mediante RANDOMIZE USR 24169 y *no* con USR 24001 que es el comienzo de DATA.

LINEA 280 DEL PROGRAMA EN BASIC

Esta es una simple sentencia de impresión y es fácil de convertir en código máquina utilizando la rutina para imprimir con DATA de la ROM (203Ch – 8252d) que ya se estudió en el Capítulo 1. Así, DATA = 5E77h (24183d) a 5EA1h (2422 5d). Se salta a ella mediante la instrucción "JR 5EA2h" desde la direc-

ción de memoria 5E75/6h. La rutina para imprimir con DATA = 5EA2h(24226d) a 5EAFh.

LINEA 29Ø Y 295 DEL PROGRAMA EN BASIC CODIGO MAQUINA DESDE 5EBØ A SEBE

En esta rutina esperamos hasta que se produce una nueva pulsación de tecla y se examina la tecla para ver si es "y" (la orden HALT no es absolutamente necesaria). Si la tecla pulsada no es "y", entonces pasamos por alto la instrucción saltando (con NZ) a 6Ø8Ah(24714d).

LINEA 3ØØ : 1 CLS DEL PROGRAMA EN BASIC CODIGO MAQUINA DESDE 5EBF a 5EC6

Esta rutina es sencilla y se habló de ella en el Capítulo 1.

LINEAS 3ØØ:2 A 33Ø DEL PROGRAMA EN BASIC

De nuevo se lleva a cabo esto utilizando las rutinas de almacenamiento de DATA y las de impresión de series con PRINT. El primer grupo de DATA se guarda en 5ECA hasta 5F72 con la rutina PRINT DATA en 5F73 hasta 5F83. El segundo grupo de DATA se guarda desde 5F84 hasta 5FF9 y la rutina PRINT DATA desde 5FFA hasta 6ØØ2. Los DATA podrían haberse agrupado con una sola rutina PRINT DATA.

La rutina PAUSE Ø va desde 6ØØ3 hasta 6ØØC; observe que el bit 5 (IY + 1) tiene que ponerse a cero desde la rutina PAUSE Ø previa. Intente sacar RES 5 (IY + 1) y "vea" el resultado.

LINEA 37Ø: 1 CLS DEL PROGRAMA EN BASIC RUTINA DESDE 6ØØE A 6Ø12

Utiliza los métodos alternativos para borrar la pantalla completa ingresando ceros con POKE en todas las direcciones de la pantalla (sin afectar a los ATTRIBUTES).

LINEAS 37Ø: 2 CLS HASTA 39Ø DEL PROGRAMA EN BASIC

DATA para rutina de impresión = 6Ø16 a 6Ø75
Rutina para imprimir datos con PRINT = 6Ø76 a 6Ø7E.
A continuación viene de nuevo la rutina PAUSE Ø.

LINEA 400 DEL PROGRAMA EN BASIC

Utilizando la rutina de establecimiento del color de BORDER y de establecimiento de atributos de color permanentes como en el Capítulo 1, podemos convertir esta línea en la rutina que va desde 608A a 609D.
La llamada CLS pone los colores en la pantalla como en el BASIC.

LINEA 410 DEL PROGRAMA EN BASIC

Esta línea establece el valor inicial de la puntuación alta. Supondremos que el valor no será mayor que 65535 y como tal utiliza los dos primeros bytes de la memoria intermedia de la impresora para guardar su valor (direcciones 5B0001-23296/7). Se puede poner a cero fácilmente con LD HL,0000h y LD (5B00),HL como se muestra en la rutina de 609C a 60A1.

LINEA 420 DEL PROGRAMA EN BASIC

DATA para impresión = 60A4 a 60C8.
Rutina para imprimir DATA con PRINT = 60C9 a 60D6

LINEA 430 DEL PROGRAMA EN BASIC

Inicializa los valores de "vidas", "puntuación" y "lugares a salvo". Las vidas pueden contarse con un byte, la puntuación con dos bytes, y los lugares a salvo con un byte. Emplearemos de nuevo la memoria intermedia de la impresora para almacenar esos valores. La dirección 5B02 contiene "vidas", las 5B03/4 contienen la "puntuación", y la 5B05 contiene "lugares a salvo". Estos valores iniciales se almacenan utilizando la rutina de 60D7 a 60E5.

LINEA 440 DEL PROGRAMA EN BASIC

Esta línea reinicializa a cero la velocidad colocando las instrucciones RET en los lugares correctos en la subrutina LINE ROLL. Rutina: 60E6 a 60F0.

LINEA 450 DEL PROGRAMA EN BASIC

DATA para impresión = 60F4 a 6128.
Rutina para imprimir DATA con PRINT = 6129 a 6136.

LINEA 455 DEL PROGRAMA EN BASIC

Se hace una prueba para ver si los lugares a salvo < > Ø; es decir, si la visualización de pantalla tiene que imprimirse o dejarse como está, borrando solamente los lugares a salvo. Necesitamos ir a por el valor almacenado en "lugares a salvo" 5BØ5 y comprobar que no es cero utilizando "AND A". Si esto no pone a uno la bandera de cero, entonces saltamos a la rutina de visualización PRINT. Si la bandera de cero se pone a uno, omitimos la rutina de visualización PRINT y saltamos al juego en sí.
Rutina: 6137 a 614Ø.

LINEAS 46Ø A 64Ø DEL PROGRAMA EN BASIC

Esta es la rutina más larga del programa, y sólo demuestra realmente lo tendioso que es imprimir una visualización.
DATA para impresión de 6141 a 63D1
Rutina para PRINT DATA 63D2 a 63DF

LINEA 65Ø DEL PROGRAMA EN BASIC

Se dará cuenta de que esta rutina PRINT contiene la variable "vidas" de modo que antes de que pueda usar la rutina de imprimir datos con PRINT, el valor correcto de "vidas" debe colocarse en DATA para imprimirse. Esto se logra utilizando la rutina de 63EØ a 63E9 que sólo recoge el valor "vidas" desde 5BØ2, le suma después 30h (48d) para convertirlo al valor ASCII correcto y finalmente lo guarda en DATA con la instrucción POKE, en la dirección 64Ø1.
DATA para imprimir 63EA a 64ØF
Rutina para PRINT DATA 641Ø a 641D

LINEA 66Ø DEL PROGRAMA EN BASIC

Esta línea establece los valores iniciales de la posición x1, y1 de impresión del hombre en la pantalla, y la posición "nueva" usada en la rutina SCREEN$ x2, y2. De nuevo utilizaremos la memoria intermedia de la impresora para guardar estos valores.

x1 = 5BØ6
y1 = 5BØ7
x2 = 5BØ8
y2 = 5BØ9

Esto se realiza fácilmente en código máquina mediante LD HL,1Ø14h:LD(5BØ6), HL : LD(5BØ8),HL. Por lo anterior observará que usar números hexadecimales es mucho más fácil que usar los decimales. Está bastante claro que con LD HL, 1Ø14h contendrá 10h (16d) y L contendrá 14 (20d). Rutina de 641E a 6426.

LINEA 67Ø DEL PROGRAMA EN BASIC

Puesto que vamos imprimiendo un carácter cuya posición depende de las variables x1, y1, es mucho más fácil utilizar LD A,n : RST 10h que una rutina PRINT DATA. Rutina de 6427 a 6445.

LINEA 68Ø DEL PROGRAMA EN BASIC

Rutina de 6446 a 6448. Dejaremos la rutina ROLL en la dirección 7E27 (32295).

LINEA 69Ø DEL PROGRAMA EN BASIC

Utilizaremos la rutina descrita en el Capítulo 8 para encontrar SCREEN$ (x2, y2), después desapilaremos el "valor" en los registros A,B,C,D,E. El carácter puede encontrarse entonces mediante LD A,(DE) y compararse con 20h (32). Recuerde que debemos borrar el espacio de trabajo antes de continuar con el programa. Rutina de 6449 a 645D. Si el carácter no es un espacio, saltamos a la rutina HIT que empieza en la dirección 6548h. El lazo principal del BASIC salta a la línea 88Ø.

LINEA 88Ø DEL PROGRAMA EN BASIC

Aquí comprobamos que el hombre no está en la línea superior de la pantalla, es decir, en "lugar a salvo". En código máquina obtenemos el valor x2 mediante LD A(5BØ8) y lo comparamos con cero. Si es cero, saltamos a la rutina de lugar seguro en la dirección 66A6. Rutina 645E a 6465. El lazo principal del BASIC salta a la línea 1Ø5Ø.

LINEA 1Ø5Ø DEL PROGRAMA EN BASIC

Como estamos imprimiendo un carácter usando dos variables x2, y2, utilizaremos LD A,n: RST 10h (como en la línea 67Ø). Rutina de 6466 a 6484.

LINEA 1060 DEL PROGRAMA EN BASIC

Se trata de recoger el valor almacenado en 5B08/9 y colocarlo en 5B06/7. Rutina de 6485 a 648A.

LINEA 1070 DEL PROGRAMA EN BASIC

Ahora cambiaremos el BASIC para mejorar el juego. En vez de utilizar la variable LAST KEY para leer el teclado, lo leeremos directamente empleando la orden IN A,(C) y así se podrán leer todas las teclas y se podrán hacer movimientos diagonales.

Para leer la tecla "1" necesitamos leer media fila de 1 a 5 y, como se describió en el Capítulo 5, la rutina que se requiere es 648B a 6493.

LINEA 1080 DEL PROGRAMA EN BASIC

Puesto que no es importante la duración y el tono del BEEP, no necesitamos convertirla exactamente al código máquina. Podemos usar la rutina BEEP, call 03B5(949d), en vez de utilizar la calculadora. Rutina de 6494 a 649C.

LINEA 1090 DEL PROGRAMA EN BASIC

Aquí actualizamos la posición de línea (x2) del hombre, de modo que recogemos el valor de 5B08, lo decrementamos dos veces, y volvemos a poner el nuevo valor en 5B08. Se incrementa la puntuación recogiendo el valor de 5B03/4, sumándole cinco y poniendo otra vez el resultado en 5B03/4. Rutina de 649D a 64AF. Para imprimir la puntuación utilizaremos las rutinas STACK BC y PRINT VALUE ON STACK, pero primero definiremos los parámetros de "PRINT AT". Rutina de 64B1 a 64C8.

LINEA 1100 DEL PROGRAMA EN BASIC

De nuevo, utilizando la instrucción IN A,(C) podemos leer el teclado, y actualizar el valor de y2(5B09). Rutina de 64C9 a 64F1. He añadido una rutina para examinar si se pulsa la tecla "6" para obtener la posibilidad de volver al BASIC en este punto. (Esto es necesario para guardar con SAVE el programa una vez que funcione correctamente). Rutina de 64F2 a 64F4.

Esto completa el lazo principal, pero encontrará que, si se ejecuta este programa, será tan rápido que hará casi invisible al hombre y parpadeará entre dos rodamientos de pantalla. Por

ello necesitamos un lazo de retardo para mantener quieto el hombre durante un momento, según se sugirió en el Capítulo 6. En vez de nuestro lazo normal podemos añadir un efecto de sonido de ruido de tráfico mediante la rutina de 64F5 a 65ØB. El JP 6427 final nos hará regresar al comienzo del lazo principal (línea 67Ø).

Rutina de choque

LINEA 7ØØ DEL PROGRAMA EN BASIC

Aquí presentamos otra variable a = x2. Utilizaremos la dirección 5CBØ para guardar este valor (una dirección poco usada en las variables del sistema). La variable "b" del lazo FOR/NEXT puede reemplazarse por LD B,19h (25d). Este valor se coloca después en la dirección 5CB1 para utilizarse en la rutina siguiente. Rutina 6548 a 6561. La llamada GOSUB 25 se reemplaza por 65ØF.

LINEA 25 DEL PROGRAMA EN BASIC

Esta se puede convertir fácilmente usando la calculadora para apilar .01.b —a simplemente con LD HL,(5CBØ) : LDA,H : SUBL; los valores a y b se almacenan consecutivamente en 5CBØ/B1. Después usamos el valor almacenado en "A" con la rutina 2D28 y la rutina BEEP 03F8. Rutina de 65ØF a 6521.

LINEA 3Ø DEL PROGRAMA EN BASIC

Este es otro caso en el que usa LD A,n : RST 1Øh ya que hay dos variables en la rutina PRINT. Rutina de 6522 a 6547.

LINEA 73Ø DEL PROGRAMA EN BASIC

La variable a está ya almacenada en 5CBØ, por lo tanto podemos establecer un lazo tomando este valor, e incrementarlo en dos después de cada paso hasta que el valor sea 14h(20d). Rutina de 6562 a 6573.

LINEA 74Ø DEL PROGRAMA EN BASIC

Aquí obtenemos el valor de "vidas" de la dirección 5BØ2, lo reducimos en uno, y volvemos a ingresar de nuevo el valor en

5BØ2. Para imprimir este valor establecemos los parámetros de PRINT AT, obtenemos después el valor, sumamos 30h(48d) para convertirlo al código ASCII, y lo imprimimos. Rutina de 6574 a 658E.

LINEA 75Ø DEL PROGRAMA EN BASIC

Rutina de 658F a 6593. Lleva consigo la reinicialización del valor x2(5BØ8) a 14h(20d).

LINEA 76Ø DEL PROGRAMA EN BASIC

Rutina de 6594 a 659A. Esta es bastante sencilla. Si "vidas" no es igual a cero, saltamos a 6446, es decir, regresamos al lazo principal.

Rutina de fin de juego

Reúne la obtención del valor de alta puntuación y puntuación y comprueba cuál es el mayor. Se hace fácilmente utilizando SBC HL, DE y examinando después el estado de la bandera CARRY (acarreo). Rutina de 659B a 65A6. Si la puntuación alta es mayor que la puntuación, saltamos a 6611.

LINEA 78Ø DEL PROGRAMA EN BASIC

Transfiere la puntuación en 5BØ3/4 a la alta puntuación en 5BØ/Ø1. Rutina de 65A7 a 65AC. Para imprimir la puntuación alta utilizamos las rutinas STACK VALUE IN BC y PRINT TOP VALUE ON CALCULATOR STACK habiendo establecido primero los parámetros PRINT AT. Rutina de 65AD a 66C2.

LINEAS 79Ø A 8ØØ DEL PROGRAMA EN BASIC

DATA para imprimir =65C5 a 661Ø.
Rutina PRINT DATA = 6611 a 6619.

LINEA 84Ø A 85Ø DEL PROGRAMA EN BASIC

Aquí usamos la variable LAST KEY para esperar una pulsación de tecla y entrar en acción de acuerdo con esto. Rutina de

661A a 6632. Si la tecla pulsada no es "y" (79h) o "n" (6Eh) saltamos de nuevo al lazo WAIT.

LINEA 860 DEL PROGRAMA EN BASIC

Esta se lleva a cabo usando LD A,n : RST 10h seguida de JP 60C9 para volver al lazo principal reimprimiéndole la visualización, pero dejando inalterada la "puntuación alta". Rutina de 6633 a 6655.

Rutina de lugar a salvo

LINEA 890 DEL PROGRAMA EN BASIC

Aquí usamos la rutina POINT DATA pero primero ponemos en DATA los valores requeridos de x1, y1 y x2, y2.
DATA para PRINT de 669A a 66A5
SET valor de x1, y1, x2, y2 de 66A6 a 66B1
Rutina PRINT DATA de 66B2 a 66BF

LINEA 900 DEL PROGRAMA EN BASIC

Esta por supuesta es innecesaria en código máquina, ya que el comienzo de DATA no necesita reinicializarse.

LINEAS 910 A 920 DEL PROGRAMA EN BASIC

FANFARE BEEP call 666A
Se trató detalladamente esta rutina en el Capítulo 7 y por ello no necesita más comentarios. Rutina de 66C0 a 66C2.

LINEAS 930 DEL PROGRAMA EN BASIC

Otro grupo de rutinas para recoger los valores actuales de "lugares a salvo" y "puntuación" y actualizarlos; viene seguida por una rutina para imprimir la puntuación utilizando las rutinas STACT VALUE IN BC y PRINT TOP VALUE ON STACK. Rutina de actualización de 66C3 a 66D3. Rutina de impresión de puntuación de 66D4 a 66E9.

LINEA 950 DEL PROGRAMA EN BASIC

Esto es una prueba para ver si los cuatro lugares seguros están ocupados, y es más fácil de programar en código máquina que

en BASIC. Rutina de 66EA a 66F1. Puede ver que primero hacemos LD A,(5BØ5) es decir, contamos los "lugares a salvo" y enmascaramos entonces el valor con AND Ø3h. Si nos queda una respuesta distinta de cero, S2 el valor original de "lugares a salvo", no era múltiplo de cuatro. Si el resultado es distinto de cero volvemos a saltar el lazo principal con JP 641E.

LINEA 96Ø A 98Ø DEL PROGRAMA EN BASIC

Estas rutinas se utilizan para incrementar la velocidad del juego cuando "lugares a salvo" = 4,8 y 12. Esto se hace quitando las instrucciones RETURN en la subrutina ROLL, permitiendo que ciertas líneas sean cambiadas por un pixel extra: línea 96Ø = 66F2 a 66FF, línea 97Ø = 67ØØ a 67Ø9, y línea 98Ø = 67ØA a 6712.

LINEA 985 DEL PROGRAMA EN BASIC

Ahora comprobamos que "lugares a salvo" no es mayor que 36. Si es así, entonces se pierde la rutina "ADD EXTRA SPIDER" ("añadir una araña extra") y volvemos al lazo principal. Restamos simplemente 37 del valor de "lugares a salvo" y si la bandera CARRY (acarreo) no está a uno (es decir lugares a salvo = 37), entonces hacemos JP NC, 6129. Rutina de 6714 a 6718.

LINEA 99Ø DEL PROGRAMA EN BASIC

Esta rutina fue tratada en el Capítulo 4 y no necesita más explicación, excepto anotar que en este caso utilizamos el valor SEED modificado en BC, y enmascaramos 1Fh (31d) : LD A,C : AND 1Fh; después le restamos dos y examinamos si se pone a uno la bandera de acarreo. Este método fue elegido porque no era necesario un verdadero número aleatorio. Rutina de 6719 a 6740.

LINEA 1ØØØ DEL PROGRAMA EN BASIC

Rutina 6741.

Todo lo que se necesita es incrementar el valor de "a".

LINEA 1ØØ5 DEL PROGRAMA EN BASIC

Se hace una prueba para ver si el valor de "a" no es mayor que 31, es decir, fuera de la pantalla (1Ø,a) se ha borrado utilizando

la rutina SCREEN$. Esto supone colocar el valor "a" en el registro B y 0ah (10d) en el registro C, llamando a las rutinas STACK VALUE IN BC Y SCREEN$, cambiando después los parámetros de la pila a los registros A,B,C,D,E y comprobando finalmente que SCREEN$ (1Ø,a) = 20h(32d). Si todo es satisfactorio, entonces obtenemos el valor final de "a" (POP AF), y se continúa. Si la rutina SCREEN$ indica que la posición no está borrada, entonces hacemos JR 6741 y así pasamos a la siguiente posición de pantalla. Rutina de 6747 a 675E.

LINEA 1Ø2Ø DEL PROGRAMA EN BASIC

Es una repetición de la rutina anterior excepto en que el valor de "a" se incrementa antes de la rutina SCREEN$. Rutina de 675F a 6776. Observe que el valor de "a" se guarda antes de acceder a la rutina SCREEN$ (676Ø PUSH AF) para usarse en la rutina siguiente.

LINEA 1Ø3Ø DEL PROGRAMA EN BASIC

Se vuelve a sacar, con POP, el valor de a, se coloca en AF y se decrementa. Este se coloca después en PRINT DATA, y se llama a la rutina PRINT. Rutina de 6777 a 6792.

LINEA 1Ø35 DEL PROGRAMA EN BASIC

Esta es una repetición de la FANFARE BEEP y sólo requiere CALL 666A. Rutina de 6793 a 6795.

LINEA 1Ø4Ø DEL PROGRAMA EN BASIC

Finalmente volvemos a saltar al lazo principal utilizando JP 6129. Rutina de 6796 a 6798.

De lo anterior puede comprobar que es realmente muy sencillo convertir un programa en BASIC a código máquina *si lo sigue paso a paso* y planea su programa con el código máquina en mente.

Un último punto, que sin duda habrá observado, es que es mejor trabajar en hexadecimal que en decimal puesto que por ejemplo, muchas de las rutinas requieren que se cargue BC con los parámetros línea/columna. Eche otro vistazo a la subrutina LEFT/RIGHT ROLL. Las direcciones de pantalla en decimal dicen muy poco, no tienen modelo aparente, pero si observa sus equivalentes en decimal verá algún tipo de modelo, obtenién-

dose una indicación más clara de lo que realmente está sucediendo.

Todo lo que se necesita ahora es que usted invente su propio programa original y brillante y empieze a convertirlo al código máquina. Le deseo buena suerte y espero que disfrute de muchas horas agradables programando sin que se le descontrolen demasiados programas.

```
org 42ØØØ 24ØØ1

5DC1h
defb $of $12 $22 $7f $ff $ff $28
defb $10
defb $80 $40 $20 $fe $fe $ff $28
defb $10
defb $7f $7f $7f $7f $f $ff $15
defb $08
defb $ff $fe $fe $fe $ff $ff $40
defb $80
defb $00 $f8 $c4 $c4 $fe $fe $28
defb $10
defb $18 $18 $24 $7e $3c $5a $a5
defb $42
defb $38 $28 $92 $72 $38 $38 $28
defb $6c
defb $01 $02 $4 $7f $7f $ff $14
defb $08
defb $f0 $48 $44 $fe $ff $ff $14


defb $Ø8
defb $ØØ $1f $23 $23 $7f $7f $14
defb $08
defb $7f $7f $7f $7f $ff $ff $Ø2
defb $Ø1
defb $fe $fe $fe $fe $fe $ff $a8
defb $1Ø
defb $10 $29 $c7 $ØØ $26 $ØØ $ØØ
defb $ØØ
defb $ØØ $44 $ff $44 $44 $ff $44


defb $ØØ
defb $ØØ $22 $55 $8f $97 $a3 $aØ
defb $ØØ
defb $ØØ $44 $aa $f1 $e9 $c5 $Ø5
defb $ØØ
defb $1Ø $1Ø $1Ø $fe $3f $1f $Øf
defb $Ø7
defb $ØØ $ØØ $ØØ $ØØ $1e $ff $ff
defb $ff
defb $6Ø $7c $54 $78 $7f $ff $fe
defb $7c


defb $ØØ $ØØ $Ø3 $Ø2 $Øf $3f $ff
defb $ØØ
defb $Ø6 $Øc $98 $fØ $eØ $55 $ff
defb $ØØ


5E69h
24169 ED 5B 7B 5C ld de,($5c7b)
24173 21 C1 5D         ld hl,$5dc1
24176 Ø1 AB ØØ         ld bc,$ØØa8
24179 ED BØ            ldir
24181 18 2B            jr +$2b


5E77h
defb $16 $Øb $Ø3 $44 $6f $2Ø $79
defb $6f
defb $75 $2Ø $77 $61 $6e $74 $2Ø
defb $69


defb $6e $73 $74 $72 $75 $63 $74
defb $69
defb $6f $6e $73 $3f $16 $Ød $Øb
defb $28
defb $79 $29 $65 $73 $16 $Øf $Øb
defb $2B
defb $6e $29 $6f


5EA2h
24226 3E Ø2            ld a,$Ø2
24228 CD Ø1 16         call $16Ø1
24231 11 77 5E         ld de,$5e77
24234 Ø1 2B ØØ         ld bc,$ØØ2b
24237 CD 3C 2Ø         call $2Ø3c
2424Ø 76               hal t
24241 FD CB Ø1 6E      bit 5,(iy+$Ø1)
24245 28 F9            jr z,–$Ø7
24247 3A Ø8 5C         ld a,($5cØ8)
2425Ø FE 79            cp $79


24252 C2 8A 6Ø         jp nz,$6Ø8a


5EBFh
24255 3E Ø2            ld a,$Ø2
24247 CD Ø1 16         call $16Ø1
2426Ø CD 6B ØD         call $Ød6b
24263 C3 73 5F         jp $5f73


5ECA
defb $16 $ØØ $Øb $4f $42 $4a $45
defb $43
defb $54 $Ød $Ød $54 $6f $2Ø $67
defb $75
deb $69 $64 $65 $2Ø $61 $2Ø $96
defb $2Ø
defb $61 $63 $72 $6f $73 $73 $2Ø
defb $61
```

```
defb $2Ø $72 $6f $61 $64 $2Ø $61
defb $6e
defb $64 $2Ø $61 $72 $69 $76 $65
defb $72
defb $2c $61 $76 $6f $69 $64 $69
defb $6e
defb $67 $2Ø $9Ø $91 $2Ø $92 $93
defb $94
defb $2Ø $95 $2Ø $aØ $a1 $a1 $a2
defb $2Ø
defb $a3 $a4 $Ød $Ød $41 $2Ø $9e
defb $9f
defb $2Ø $7Ø $61 $74 $72 $6f $6c
defb $73
defb $2Ø $74 $68 $65 $2Ø $63 $65
defb $6e
defb $74 $72 $61 $6c $2Ø $69 $73
defb $6c
defb $61 $6e $64 $2e $Ød $Ød $54
defb $68
defb $65 $72 $65 $2Ø $61 $72 $65


defb $2Ø
defb $34 $2Ø $48 $4f $4d $45 $53
defb $2Ø
defb $74 $6f $2Ø $62 $65 $2Ø $66
defb $69
defb $6c $6c $65 $64 $2e $2Ø $69
defb $2e
defb $65 $2e $2Ø $67 $61 $7Ø $73
defb $2Ø
defb $69 $6e $2Ø $66 $65 $6e $63
defb $65
defb $2Ø $9d $9d $9d $2Ø $9d $9d
defb $9d $Ød
```

```
5F73h
24435 3E Ø2            ld a,$Ø2
24437 CD Ø1 16         call $16Ø1
2444Ø 11 CA 5E         ld de,$5eca
24443 Ø1 A9 ØØ         ld bc,$ØØa9


24446 CD 3C 2Ø         call $2Ø3c
24449 C3 FA 5F         jp $5ffa
```

```
5F84h
defb $Ød $4f $6e $63 $65 $2Ø $61
defb $6c
defb $6c $2Ø $34 $2Ø $48 $4f $4d
defb $45
defb $53 $2Ø $61 $72 $65 $2Ø $66
defb $69
defb $6c $6c $65 $64 $2Ø $74 $68
defb $65
defb $2Ø $73 $7Ø $65 $65 $64 $2Ø
defb $77
defb $69 $6c $6c $2Ø $69 $6e $63
defb $72
defb $65 $61 $73 $65 $2c $2Ø $61
defb $6e
defb $2Ø $65 $78 $74 $72 $61 $2Ø
defb $65
defb $2Ø $61 $64 $64 $65 $64 $2Ø
defb $61
defb $6e $64 $2Ø $74 $68 $65 $2Ø
defb $48
defb $4f $4d $45 $53 $2Ø $77 $69
defb $6c
defb $6c $65 $6d $7Ø $74 $79 $16
defb $12
defb $Ø9 $5Ø $72 $65 $73 $73 $2Ø
defb $61
defb $6e $79 $2Ø $6b $65 $79
```

```
5FFAh
2457Ø 11 84 5F         ld de,$5f84
24573 Ø1 76 ØØ         ld bc,$ØØ76
24576 CD 3C 2Ø         call $203c
24579 FD CB Ø1 AE      res 5, (iy+Ø1
24583 76               halt


24584 FD CB Ø1 6E      bit 5,(iy+$Ø1
24588 28 F9            jr z,–$Ø7
```

```
6ØØEh
2459Ø Ø6 18            ld b,$18
24592 CD 44 ØE         call $Øe44
24595 C3 76 6Ø         jp $6Ø76
```

```
6Ø16h
defb $16 $Ø7 $Øb $43 $4f $4e $54
defb $52
defb $4f $4c $53 $0d $Ød $2Ø $2Ø
defb $2Ø
defb $2Ø $2Ø $2Ø $5e $2Ø $2Ø $2Ø


defb $2Ø
defb $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø
defb $2Ø


defb $2Ø $2Ø $2Ø $3c $3Ø $3e $12
defb $Ø1
defb $16 $Øb $Ø6 $31 $12 $ØØ $2Ø
defb $32
defb $2Ø $33 $2Ø $34 $2Ø $35 $2Ø
defb $36
defb $2Ø $37 $2Ø $38 $3Ø $12 $Ø1
defb $39
defb $12 $ØØ $2Ø $12 $Ø1 $3Ø $12
defb $ØØ
defb $16 $12 $Ø5 $5Ø $72 $65 $73
defb $73
defb $2Ø $61 $6e $79 $2Ø $6b $65
defb $79
defb $2Ø $74 $6f $2Ø $5Ø $4c $41
defb $59
```

```
6Ø76h
24694 11 16 6Ø          ld de,$6Ø16


24697 Ø1 6Ø ØØ          ld bc,$ØØ6Ø
247ØØ CD 3C 2Ø          call $2Ø3c
247Ø3 FD CB Ø1 AE       res 5, (iy+$Ø1
247Ø7 76                hal t
247Ø8 FD CB Ø1 6E       bit 5,(iy+$Ø1
24712 28 F9             jr z,-$Ø7


6Ø8Ah
24714 3E Ø5             ld a,$Ø5
24716 CD 9B 22          call $229b
24719 3E 68             ld a,$68
24721 32 8D 5C          ld ($5c8d),a
24724 3E Ø2             ld a,$Ø2
24726 CD Ø1 16          call $16Ø1
24729 CD 6B ØD          call $Ød6b


6Ø9Ch


24732 21 ØØ ØØ          ld hl,$ØØØØ
24735 22 ØØ 5B          ld ($5bØØ),hl
24738 18 25             jr +$25


6ØA4h
defb $11 $Ø4 $16 $Øa $ØØ $2Ø $2Ø
defb $2Ø
defb $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø
defb $2Ø
defb $2Ø $2Ø $2Ø $2Ø $9e $9f $2Ø
defb $2Ø


defb $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø
defb $2Ø
defb $2Ø $2Ø $2Ø $2Ø $2Ø


6ØC9h
24777 3E Ø2             ld a,$Ø2


24779 CD Ø1 16          call $16Ø1
24782 11 A4 6Ø          ld de,$6Øa4
24785 Ø1 25 ØØ          ld bc,$ØØ25
24788 CD 3C 2Ø          call $2Ø3c


6ØD7h
24791 3E Ø9             ld a,$Ø9
24793 32 Ø2 5B          ld ($5bØ2),a
24796 21 ØØ ØØ          ld hl,$ØØØØ
24799 22 Ø3 5B          ld ($5bØ3), hl
248Ø2 AF                xor a
248Ø3 32 Ø5 5B          ld ($5bØ5),a

6ØE6h
248Ø6 3E C9             ld a,$c9
248Ø8 32 A9 7E          ld ($7ea9),a
24811 32 C2 7E          ld ($7ec2),a


24812 32 D5 7E          ld ($7ed5),a
24817 C3 29 61          jp $6129


6ØF4h
defb $16 $ØØ $ØØ $11 $Ø4 $9d $9d
defb $9d
defb $9d $11 $Ø7 $2Ø $11 $Ø4 $9d
defb $9d
defb $9d $9d $9d $9d $11 $Ø7 $2Ø
defb $11
defb $Ø4 $9d $9d $9d $9d $9d $9d
defb $9d
defb $11 $Ø7 $2Ø $11 $Ø4 $9d $9d
defb $9d
defb $9d $9d $9d $9d $11 $Ø7 $2Ø
defb $11
defb $Ø4 $9d $9d $9d $9d


6129h
24873 3E Ø2             ld a,$Ø2
24875 CD Ø1 16          call $16Ø1
24878 11 F4 6Ø          ld de,$6Øf4
24881 Ø1 35 ØØ          ld bc,$ØØ35
24884 CD 3C 2Ø          call $2Ø3c
6137h
24887 3A Ø5 5B          ld a,($5bØ5)
2489Ø A7                and a
24891 C2 1E 64          jp nz,$641 e
24894 C3 D2 63          jp $63d2


6141h
defb $11 $Ø4 $1Ø $Ø5 $8c $8c $8c
defb $8c
defb $8c $8c $8c $8c $8c $8c $8c
defb $8c
defb $8c $8c $8c $8c $8c $8c $8c
defb $8c
defb $8c $8c $8c $8c $8c $8c $8c
defb $8c
defb $8c $8c $8c $8c $11 $Ø5 $1Q
defb $ØØ
defb $2Ø $aØ $a1 $a1 $a2 $2Ø $2Ø
defb $2Ø
defb $aØ $a1 $a1 $a1 $a2 $2Ø $2Ø
defb $2Ø
defb $2Ø $aØ $a1 $a1 $a2 $2Ø $2Ø
defb $2Ø
defb $aØ $a1 $a1 $a2 $2Ø $2Ø $2Ø
defb $2Ø
defb $1Ø $Ø7 $2Ø $2Ø $2Ø $9c $9c
defb $9c
defb $2Ø $2Ø $2Ø $2Ø $2Ø $9c $9c
defb $9c
defb $9c $9c $2Ø $2Ø $2Ø $2Ø $2Ø
```

```
defb $9c
defb $9c $9c $2Ø $2Ø $2Ø $2Ø $2Ø
defb $2Ø
defb $9c $1Ø $Ø2 $2Ø $2Ø $a3 $a4
defb $2Ø
defb $2Ø $2Ø $2Ø $2Ø $a3 $a4 $2Ø
defb $2Ø
defb $2Ø $2Ø $2Ø $a3 $a4 $2Ø $2Ø
defb $2Ø
defb $2Ø $a3 $a4 $2Ø $2Ø $2Ø $2Ø
defb $2Ø
defb $2Ø $a3 $a4 $2Ø $1Ø $Ø7 $9c
defb $9c
defb $2Ø $2Ø $2Ø $9c $2Ø $2Ø $2Ø
defb $2Ø
defb $2Ø $2Ø $9c $9c $9c $9c $9c
defb $2Ø
defb $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø
defb $9c
defb $9c $9c $2Ø $2Ø $2Ø $2Ø $1Ø
defb $Ø1


defb $a1 $a2 $2Ø $2Ø $2Ø $2Ø $2Ø
defb $aØ
defb $a1 $a1 $a2 $2Ø $2Ø $2Ø $2Ø
defb $2Ø
defb $2Ø $aØ $a1 $a1 $a1 $a1 $a2
defb $2Ø
defb $2Ø $2Ø $2Ø $2Ø $aØ $a1 $a1


defb $a1
defb $1Ø $Ø7 $2Ø $2Ø $2Ø $9c $9c
defb $2Ø
defb $2Ø $9c $9c $9c $9c $9c $2Ø
defb $2Ø
defb $2Ø $9c $9c $2Ø $2Ø $2Ø $2Ø
defb $2Ø
defb $2Ø $9c $9c $9c $2Ø $2Ø $2Ø
defb $2Ø
defb $2Ø $2Ø $1Ø $ØØ $a4 $2Ø $2Ø
defb $2Ø
defb $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø $a3
defb $a4
defb $2Ø $2Ø $2Ø $2Ø $2Ø $a3 $a4
defb $2Ø
defb $2Ø $2Ø $2Ø $2Ø $2Ø $a3 $a4
defb $2Ø
defb $2Ø $2Ø $2Ø $a3 $11 $Ø4 $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $11
defb $ØØ
defb $1Ø $Ø7 $16 $Ød $ØØ $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d


defb $9d $9d $9d $9d $9d $1Ø $Ø3
defb $2Ø
defb $9Ø $91 $2Ø $95 $2Ø $2Ø $9Ø
defb $91
defb $2Ø $2Ø $92 $93 $92 $93 $94
defb $2Ø
defb $95 $2Ø $95 $2Ø $2Ø $2Ø $92
defb $93
defb $94 $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø
defb $1Ø
defb $Ø7 $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $1Ø $Ø5 $2Ø $9Ø $91 $2Ø
defb $2Ø
defb $2Ø $2Ø $9Ø $91 $2Ø $2Ø $20


defb $2Ø
defb $9Ø $91 $2Ø $2Ø $9Ø $91 $2Ø
defb $2Ø
defb $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø $9Ø


defb $91
defb $2Ø $2Ø $2Ø $1Ø $Ø7 $3d $3d
defb $3d
defb $3d $3d $3d $3d $3d $3d $3d
defb $3d
defb $3d $3d $3d $3d $3d $3d $3d
defb $3d
defb $3d $3d $3d $3d $3d $3d $3d
defb $3d
defb $3d $3d $3d $3d $3d $10 $Ø4
defb $98
defb $2Ø $2Ø $2Ø $2Ø $2Ø $97 $98
defb $2Ø
defb $2Ø $2Ø $97 $98 $2Ø $2Ø $2Ø
defb $2Ø
defb $2Ø $2Ø $2Ø $97 $98 $2Ø $2Ø
defb $2Ø

defb $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø $ 97
defb $1Ø
defb $Ø7 $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $1Ø $Ø6 $9a $9b $2Ø $2Ø
defb $2Ø
defb $97 $98 $2Ø $2Ø $97 $98 $2Ø
defb $95
defb $2Ø $2Ø $95 $2Ø $2Ø $99 $9a
defb $9b
defb $9a $9b $2Ø $2Ø $2Ø $95 $2Ø
defb $2Ø
defb $2Ø $2Ø $99 $11 $Ø4 $1Ø $ØØ
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d


defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
```

```
defb $11
defb $Ø4 $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø
defb $2Ø
defb $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø
defs $2Ø
defb $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø
defb $2Ø
defb $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø
defb $2Ø $2Ø


63D2h
25554 3E Ø2          ld a,$Ø2
25556 CD Ø1 16       call $16Ø1


25559 11 41 61       ld de,$6141
25562 Ø1 91 Ø2       ld bc,$Ø291
25565 CD 3C 2Ø       call $2Ø3C


63EØh
25568 3A Ø2 5B       ld a,($5bØ2)
25571 C6 3Ø          add a,$3Ø
25573 32 Ø1 64       ld ($64Ø1),a
25576 18 26          jr +$26


63EAh
defb $11 $Ø1 $1Ø $Ø7 $2Ø $53 $43
defb $4f
defb $52 $45 $2Ø $16 $15 $Øb $2Ø
defb $4d
defb $45 $4e $2Ø $11 $Ø5 $1Ø $ØØ
defb $39


defb $11 $Ø1 $1Ø $Ø7 $2Ø $48 $49
defb $2d
defb $53 $43 $4f $52 $45 $2Ø


641Øh
25616 3E Ø2          ld a,$Ø2
25618 CD Ø1 16       call $16Ø1
25621 11 EA 63       ld de,$63ea
25624 Ø1 26 ØØ       ld bc,$ØØ26
25627 CD 3C 20       call $2Ø3c


641Fh
2563Ø 21 14 1Ø       ld hl,$1Ø14
25633 22 Ø6 5B       ld ($5bØ6),hl
25636 22 Ø8 5B       ld ($5bØ8),hl


6427h
25639 3E Ø2          ld a,$Ø2
25641 CD Ø1 16       call $16Ø1
25644 3E 11          ld a,$11
25646 D7             rst $1Ø
25647 3E Ø8          ld a,$Ø8
25649 D7             rst $1Ø
2565Ø 3E 1Ø          ld a,$1Ø
25652 D7             rst $1Ø
25653 3E Ø8          ld a,$Ø8
25655 D7             rst $1Ø
25656 3E 16          ld a,$16
25658 D7             rst $1Ø
25659 3A Ø6 5B       ld a,($5bØ6)
25662 D7             rst $1Ø
25663 3A Ø7 5B       ld a,($5bØ7)
25666 D7             rst $1Ø
25667 3E 2Ø          la d,$2Ø
25669 D7             rst $1Ø


6446h
2567Ø CD 27 7E       call $7e27


6449h
25673 ED 4B Ø8 5B    ld bc,($5bØ8)
25677 CD 38 25       call $2538
2568Ø CD F1 2B       call $2bf1
25683 1A             1d a,(de)
25684 F5             push af
25685 CD BF 16       call $16bf
25688 F1             pop af
25689 FE 2Ø          cp $2Ø
25691 C2 48 65       jp nz, $6548


645Eh


25694 3A Ø8 5B       ld a,($5bØ8)
25697 FE ØØ          cp $ØØ
25699 CA A6 66       jp z,$66a6


6466h
257Ø2 3E Ø2          ld a,$Ø2
257Ø4 CD Ø1 16       call $16Ø1
257Ø7 3E 11          ld a,$11
257Ø9 D7             rst $1Ø
2571Ø 3E Ø8          ld a,$Ø8
25712 D7             rst $1Ø
25713 3E 1Ø          ld a,$1Ø
25715 D7             rst $1Ø
25716 3E Ø8          ld a,$Ø8
25718 D7             rst $1Ø
25719 3E 16          ld a,$16
25721 D7             rst $1Ø
25722 3A Ø8 5B       ld a,($5bØ8)
25725 D7             rst $1Ø


25726 3A Ø9 5B       ld a,($5bØ9)
25729 D7             rst $1Ø
2573Ø 3E 96          ld a,$96
25732 D7             rst $1Ø


6485h
25733 2A Ø8 5B       ld hl,($5bØ8)
25736 22 Ø6 5B       ld ($5bØ6),hl


648Bh


25739 Ø1 FE F7       ld bc,$f7fe
25472 ED 78          in a,(c)
25744 CB 47          bit Ø,a
25746 2Ø 1C          jr nz,+$1c
```

```
6494h
25748 21 32 ØØ          ld hl,$ØØ32
25751 11 Ø5 ØØ          ld de,$ØØØ5
25754 CD B5 Ø3          call $Ø3b5


649Dh
25757 3A Ø8 5B          ld a,($5bØ8)
2576Ø 3D                dec a
25761 3D                dec a
25762 32 Ø8 5B          ld ($5bØ8),a
25765 2A Ø3 5B          ld hl,($5bØ3)
25768 23                inc hl
25769 23                inc hl
2577Ø 23                inc hl
25771 23                inc hl
25772 23                inc hl
25773 22 Ø3 5B          ld ($5bØ3),hl
25776 ØØ                nop
25777 3E Ø2'            ld a,$Ø2


25779 CD Ø1 16          call $16Ø1
25782 3E 16             ld a,$16
25784 D7                rst $1Ø
25785 3E 15             ld a,$15
25787 D7                rst $1Ø
25788 3E Ø7             ld a,$Ø7
2579Ø D7                rst $1Ø
25791 ED 4B Ø3 5B       ld bc,($5bØ3)
25795 CD 2B 2D          call $2d2b
25798 CD E3 2D          call $2de3


64C9h
258Ø1 Ø1 FE EF          ld bc,$effe
258Ø4 ED 78             in a,(c)
258Ø6 CB 47             bit Ø,a
258Ø8 2Ø ØE             jr nz,+$Øe
2581Ø 3A Ø9 5B          ld a,($5bØ9)
25813 FE 1F             cp $1f
25815 28 Ø1             jr z,+$Ø1


25817 3C                inc a
25818 32 Ø9 5B          ld ($5bØ9),a
25821 C3 F5 64          jp $64f5
25824 CB 4F             bit 1,a
25826 2Ø ØE             jr nz,+$Øe
25828 3A Ø9 5B          ld a,($5bØ9)
25831 FE ØØ             cp $ØØ
25833 28 Ø1             jr z,+$Ø1
25835 3D                dec a


2586 32 Ø9 5B           ld ($5bØ9),a
25839 C3 F5 64          jp $64f5


64F2h
25842 CB 67.            bit 4,a
25844 CB                ret z


64F5h
25845 Ø6 ØA             ld b,$Øa
25847 11 78 ØØ          ld de,$ØØ78
2585Ø 21 Ø1 ØØ          ld hl, $ØØØ1
25853 E5                push hl
25854 D5                push de
25855 C5                push bc
25856 CD B5 Ø3          call $Ø3b5
25859 C1                pop bc
2586Ø D1                pop de
25861 E1                pop hl
25862 7D                ld a,l
5863 3C                 inc a
25864 6F                ld l,a
25865 ØØ                nop
25866 1Ø F1             djnz –$Øf
25868 C3 27 64          jp $6427


65ØFh
25871 EF                rst $28


651Øh
defb $34 $ea $23 $d7 $Øa $3d $38


6517h
25879 2A BØ 5C          ld hl,($5cbØ)
25882 7C                ld a,h
25883 95                sub l
25884 CD 2B 2D          call $2d2b
25887 CD F8 Ø3          call $Ø3f8
2589Ø 3E Ø2             ld a,$Ø2
25892 CD Ø1 16          call $16Ø1
25895 3E 15             ld a,$15
25897 D7                rst $1Ø
25898 3E Ø1             ld a,$Ø1
259ØØ D7                rst $1Ø
259Ø1 3E 11             ld a,$11


259Ø3 D7                rst $1Ø
259Ø4 3E Ø8             ld a,$Ø8
259Ø6 D7                rst $1Ø
259Ø7 3E 1Ø             ld a,$1Ø
259Ø9 D7                rst $1Ø
2591Ø 3E Ø8             ld a,$Ø8
25912 D7                rst $1Ø
25913 3E 16             ld a,$16
25915 D7                rst $1Ø
25916 3A BØ 5C          ld a,($5cbØ)
25919 D7                rst $1Ø
2592Ø 3A Ø8 5B          ld a,($5bØ9)
25923 D7                rst $1Ø
25924 3E 96             ld a,$96
25926 D7                rst $1Ø
25927 C9                ret


6548h
25928 3A Ø8 5B          ld a,($5bØ8)
```

```
25931 32 BØ 5C      ld ($5cbØ),a
25934 Ø6 19         ld b,$19
25936 78            ld a,b
25937 C5            push bc
25938 32 B1 5C      ld ($5cb1),a
25941 CD ØF 65      call $65Øf
25944 CD ØF 65      call $65Øf
25947 C1            pop bc
25948 Ø4            inc b
25949 78            ld a,b
2595Ø FE 23         cp $23
25952 2Ø EF         jr nz,–$11


6562h
25954 CD ØF 65      call $65Øf
25957 CD ØF 65      call $65Øf
2596Ø 3A BØ 5C      ld a,($5cbØ)
25963 3C            inc a
25964 3C            inc a


25965 32 BØ 5C      ld ($5cbØ),a
25968 FE 14         cp $14
2597Ø 2Ø EE         jr nz,–$12


6574h
25972 3A Ø2 5B      ld a,($5bØ2)
25975 3D            dec a
25976 32 Ø2 5B      ld ($5bØ2),a
25979 3E Ø2         ld a,$Ø2
25981 CD Ø1 16      call $16Ø1
25984 3E 16         ld a,$16
25986 D7            rst $1Ø
25987 3E 16         ld a,$16
25989 D7            rst $1Ø
2599Ø 3E 1Ø         ld a,$1Ø
25992 D7            rst $1Ø
25993 3A Ø2 5B      ld a,($5bØ2)
25996 C6 3Ø         add a,$3Ø
25998 D7            rst $1Ø


658Fh
25999 3E 14         ld a,$14
26ØØ1 32 Ø2 5B      ld ($5bØ2),a


6594h
26ØØ4 3A Ø2 5B      ld a,($5bØ2)
26ØØ7 A7            and a
26ØØØ8 C2 46 64     jp nz,$6446


659Bh
26Ø11 AF            xor a
26Ø12 2A Ø3 5B      ld hl,($5bØ3)
26Ø15 ED 5B ØØ 5B   ld de,($5bØØ)
26Ø19 ED 52         sbc hl,de


26Ø21 38 6A         jr c,+$6a

65A7h
26Ø23 2A Ø3 5B      ld hl,($5bØ3)
26Ø26 22 ØØ 5B      ld ($5bØØ),hl
26Ø29 E5            push hl
26Ø3Ø C1            pop bc
26Ø31 CD 2B 2D      call $2d2b
26Ø34 3E Ø2         ld a,$Ø2
26Ø36 CD Ø1 16      call $16Ø1
26Ø39 3E 16         ld a,$16
26Ø41 D7            rst $1Ø
26Ø42 3E 15         ld a,$15
26Ø44 D7            rst $1Ø
26Ø45 3E 1B         ld a,$1b
26Ø47 D7            rst $1Ø
26Ø48 CD E3 2D      call $2de3
26Ø51 18 4C         jr +$4c


65C5h
defb $12 $Ø1 $11 $Ø7 $16 $Øc $ØØ
defb $2Ø
defb $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø
defb $2Ø
defb $2Ø $2Ø $47 $41 $4d $45 $2Ø
defb $2Ø
defb $4f $56 $45 $52 $2Ø $2Ø $2Ø
defb $2Ø
defb $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø $2Ø
defb $16
defb $Øe $ØØ $12 $ØØ $2Ø $2Ø $41
defb $6e
defb $6f $74 $68 $65 $72 $2Ø $67
defb $61
defb $6d $65 $2Ø $3f $2Ø $28 $79
defb $29
defb $65 $73 $2Ø $2Ø $28 $6e $29
defb $6f


defb $2Ø $2Ø $2Ø $2Ø


6611h
26129 11 C5 65      ld de,$65c5
26132 Ø1 4C ØØ      ld bc,$ØØ4c
26135 CD 3C 2Ø      call $2Ø3c


661Ah
26138 76            halt
26139 FD CB Ø1 6E   bit 5,(iy+$Ø1
26143 28 F9         jr z,–$Ø7
26145 FD CB Ø1 AE   res 5,(iy+$Ø1
26149 3A ØB 5C      ld a,($5cØb)
26152 FE 6E         cp $6e
26154 2Ø Ø3         jr nz,+$Ø3
26156 CD ØØ ØØ      call $ØØØØ
26159 FE 79         cp $79


26161 2Ø E7         jr nz,–$19
```

```
6633h
26163 3E Ø2        ld a,$Ø2
26165 CD Ø1 16     call $16Ø1
26168 3E 11        ld a,$11
2617Ø D7           rst $1Ø
26171 3E Ø5        ld a,$Ø5
26173 D7           rst $1Ø
26174 3E 16        ld a,$16
26176 D7           rst $1Ø
26177 3E 15        ld a,$15
26179 D7           rst $1Ø
2618Ø 3E Ø7        ld a,$Ø7
26182 D7           rst $1Ø
26183 3E 2Ø        ld a,$2Ø
26185 D7           rst $1Ø
26186 3E 2Ø        ld a,$2Ø
26188 D7           rst $1Ø

26189 3E 2Ø        ld a,$2Ø
26191 D7           rst $1Ø
26192 3E 2Ø        ld a,$2Ø
26194 D7           rst $1Ø
26195 C3 C9 6Ø     jp $6Øc9


6656h
defb $ØØ $ØØ $ØØ $ØØ $ØØ $ØØ $ed
defb $Øb
defb $ed $Øb $fØ $1Ø $ec $Øb $ec
defb $1Ø
defb $ec $Øb $ec $1Ø


666Ah
26218 Ø6 Ø7        ld b,$Ø7
2622Ø 21 5C 66     ld hl,$665c
26223 7E           ld a,(hl)

26225 32 78 66     ld ($6678),a
26227 23           inc hl
26228 C5           push bc
26229 E5           push hl
2623Ø EF           rst $28
defb $34 $ec $4c $cc $cc $cc $38
26238 E1           pop hl
26239 7E           ld a,(hl)
2624Ø              push hl
26241 CD 28 2D     call $2d28
26244 CD F8 Ø3     call $Ø3f8
26247 E1           pop hl
26248 C1           pop bc
26249 23           inc hl
2625Ø 1Ø E3        djnz –$1d
26252 3E Ø1        ld a,$Ø1
26254 CD 28 2D     call $2d28
26257 3E 14        ld a,$14
26259 CD 28 2D     call $2d28
26262 CD F8 Ø3     call $Ø3f8
26265 C9           ret


669Ah
defb $11 $Ø8 $1Ø $Ø8 $16 $Ø2 $Ø4
defb $2Ø
derb $16 $ØØ $Ø4 $96


66A6h
26278 2A Ø6 5B     ld hl,($5bØ6)
26281 22 9F 66     ld ($669f),hl
26284 2A Ø8 5B     ld hl,($5bØ8)
26287 22 A3 66     ld ($66a3),hl
2629Ø 3E Ø2        ld a,$Ø2
26292 CD Ø1 16     call $16Ø1
26295 11 9A 66     ld de,$669a
26298 Ø1 ØC ØØ     ld bc,$ØØØc
263Ø1 CD 3C 2Ø     call $2Ø3c
263Ø4 CD 6A 66     call $666a


66C3h
263Ø7 3A Ø5 5B     ld a,($5bØ5)
2631Ø 3C           inc a
26311 32 Ø5 5B     ld ($5bØ5),a
26314 2A Ø3 5B     ld hl,($5bØ3)
26317 11 32 ØØ     ld de,$ØØ32
2632Ø 19           add hl,de
26321 22 Ø3 5B     ld ($5bØ3),hl
26324 E5           push hl
26325 3E Ø2        ld a,$Ø2
26327 CD Ø1 16     call $16Ø1
2633Ø 3E 16        ld a,$16
26332 D7           rst $1Ø
26333 3E 15        ld a,$15
26335 D7           rst $1Ø
26336 3E Ø7        ld a,$Ø7
26338 D7           rst $1Ø
26339 C1           pop bc

2634Ø CD 2B 2D     call $2d2b
26343 CD E3 2D     call $2de3


66EAh
26346 3A Ø5 5B     ld a,($5bØ5)
26349 E6 Ø3        and $Ø3
26351 C2 1E 64     jp nz,$641e


66F2h
26354 AF           xor a
26355 3A Ø5 5B     ld a,($5bØ5)
26538 FE Ø4        cp $Ø4
2636Ø 2Ø Ø6        jr nz,+$Ø6
26362 AF           xor a,
26363 32 A9 7E     ld ($7ea9),a
26366 18 19        jr +$19


67ØØh
26368 FE Ø8        cp $Ø8
2637Ø 2Ø Ø6        jr nz,+$Ø6
26372 AF           xor a
26373 32 C2 7E     ld ($7ec2),a
26376 18 ØF        jr +$Øf
```

```
670Ah
26378 FE 0C          cp $0c
26380 20 06          jr nz,+$06
26382 AF             xor a
26383 32 D5 7E       ld ($7ed5),a
26386 18 05          jr +$05

6714h

26388 D6 25          sub $25
26390 D2 29 61       jp nc,$6129


5718h
26393 ED 4B 76 5C    ld bc,($5c76)
26397 CD 2B 2D       call $2d2b
26400 EF             rst $28


6721h
defb $a1 $0f $34 $37 $16 $04 $34
defb $80
defb $41 $00 $00 $80 $32 $02 $a1
defb $03 $31 $38


6733h


26419 CD A2 2D       call $2da2
26422 ED 43 76 5C    ld ($5c76),bc
26246 79             ld a,c
26427 E6 1F          and $1f
26429 D6 02          sub $02
26431 38 05          jr c,+$05


6741h
26433 3C             inc a


6742h
26434 FE 20          cp $20
26436 20 01          jr nz,+$01
26438 AF             xor a

6747h
26439 F5             push af
26440 47             ld b,a
26441 0E 0A          ld c,$0a
26443 CD 38 25       call $2538
26446 CD F1 2B       call $2bf1
26449 1A             ld a,(de)
26450 F5             push af
26451 CD BF 16       call $16bf
26454 F1             pop af
26455 FE 20          cp $20
26457 28 03          jr z,+$03
26459 F1             pop af
26460 18 E3          jr -$1d
26462 F1             pop af


675Fh
26463 3C             inc a
26464 F5             push af


26465 47             ld b,a
26466 0F 0A          ld c,$0a
26468 CD 38 25       call $2538
26471 CD F1 2B       call $2bf1
26474 1A             ld a,(de)
26475 F5             push af
26476 CD BF 16       call $16bf
26479 F1             pop af
26480 FE 20          cp $20
26482 28 03          jr z,+$03
26484 F1             pop af
26485 18 CA          jr -$36
6777h
26487 F1             pop af
26488 3D             dec a
26489 32 82 67       ld ($6782),a
26492 18 07          jr +$07


677Eh
defb $11 $04 $16 $0a $19 $9e $9f


6785h
26501 3E 02          ld a,$02


26503 CD 01 16       call $1601
26506 11 7E 67       ld de,$677e
26509 01 07 00       ld bc,$0007
26512 CD 3C 20       call $203c

6793h
26515 CD 6A 66       call $666a


6796h
26518 C3 29 61       jp $6129
```

Programa "Cross (código máquina)"

Apéndice 1

Reemplazamiento de rutinas de la ROM

Las rutinas en ROM de la Spectrum están escritas de forma que se pueda reconocer cualquier parámetro equivocado y obtener un informe del error. Por ejemplo, si intenta ingresar PRINT AT 24,33; "Mensaje" entonces la rutina PRINT detectará que la línea 24, columna 33 está fuera de la pantalla.

Cuando se escriben programas en código máquina se puede suponer que se utilizarán siempre los parámetros correctos y así, las rutinas en ROM pueden reescribirse sin la detección de errores, etc., lo que las hace más rápidas, y pueden estar completamente bajo el control del programador.

Los seis programas siguientes muestran como pueden reescribirse CLS., ATTR.SET, PRINT, PLOT, UNPLOT, SCR$ y POINT. La rutina PRINT maneja los caracteres del 32 al 127 inclusive (aunque podría adaptarse para imprimir cualquier carácter) y realmente lo que hace es poner el carácter en la pantalla con POKE, por lo que es innecesario emplear OPEN CHANNEL 2 y nos permitirá imprimir en las líneas 22 y 23. Las direcciones 23728/9 de las variables no usadas se emplean para guardar la posición actual de PRINT.

La rutina SCR$ detecta los caracteres del 32 al 127 en las líneas de la 0 a la 21 como en la versión del BASIC, pero también nos permite examinar las líneas 22 y 23. Las rutinas PLOT y UNPLOT tienen los mismos parámetros que en el BASIC, con 0,0 en la esquina inferior izquierda de la línea 23. (Omitiendo las instrucciones LDA,175 en las direcciones 2374/5 haremos que sean 0,0 en la esquina superior izquierda de la línea 1). Lo anterior se aplica también a la rutina POINT. La dirección a modificar es 23777. Verá que las rutinas PLOT, UNPLOT y POINT usan idénticas subrutinas FIND para encontrar el pixel a examinar; por lo tanto si usase las tres rutinas podrían compartir esta subrutina y ahorrar más memoria. Podría aumentarse todavía más la velocidad añadiendo DI (deshabilitar interrupciones) al comienzo de estas rutinas, o al principio de su programa, y EI al final (o cuando se requiera una llamada a KEYSCAN). Con estos programas como guía usted podría escribir una rutina CIRCLE (que sea más rápida que la de la Spectrum).

```
org 2376Ø
CLS (Lento)

2376Ø 21 ØØ 4Ø      ld hl,16384
L1
23763 36 ØØ         ld (hl),Ø
23765 23            inc hl
23766 7C            ld a,h
23767 FE 58         cp 88
23769 2Ø FB         jr nz, L1
23771 C9            ret

CLS (Rápido)

23772 21 ØØ 4Ø      ld hl,16384
23775 11 Ø1 4Ø      ld de,16385
23778 Ø1 FF17       ld bc,6143
23781 36 ØØ         ld (hl),Ø
23783 ED BØ         ldir
23785 C9            ret
```

Programa A1.1

```
org 2376Ø
LLENAR ATRIBUTOS

EJEMPLO
INK 2 PAPER 5
BRIGHT 1 FLASH 1

2+4Ø+64+128=234

2376Ø 3E EA         ld a,234
23762 21 ØØ 58      ld hl,22528
23765 11 Ø1 58      ld de,22529
23768 Ø1 FF Ø2      ld bc,767
23771 77            ld (hl),a
23772 ED BØ         ldir
23774 C9            ret
```

Programa A1.2

```
org 2376Ø
RUTINA IMPRIMIR SERIES
SACA (CON POKE) CARACTERES A
PANTALLA  NO UTILIZA RST 16d

DIRECCION 23728 CONTIENE COLUMNA
DIRECCION 23729 CONTIENE LINEA

2376Ø 11 E4 5C      ld de,DATA
L4
23763 1A            ld a,(de)
23764 CB 7F         bit 7,a
23766 2Ø Ø6         jr nz,L3
23768 CD Ø2 5D      call PRINT
23771 13            inc de
23772 18 F5         jr L4
L3
23774 CB BF         res 7,a
23776 CD Ø2 5D      call PRINT
23779 C9            ret


DATA
defs McGRAW-HILL BOOK Co.
defs (UK) Ltd

128+CHR 46=",,"

defb 174

RUTINA IMPRIMIR

OBTENER LINEA/COLUMNA EN HL

IMPRIMIR
2381Ø D5            push de
23811 2A BØ 5C      ld hl,(23728)
23814 E5            push hl
23815 F5            push af

EXAMINAR SI SE SALE DE PANTALLA

23816 AF            xor a

23817 7C            ld a,h
23818 D6 18         sub 24
2382Ø 38 Ø2         jr c,OK
23822 CF            rst 8
defb 4

ENCONTRAR POSICION PANTALLA

DE ACUERDO
23824 11 ØØ 4Ø      ld de,16384
23827 19            add hl,de
23828 7C            ld a,h
23829 E6 Ø7         and 7
23831 ØF            rrca
23832 ØF            rrca
23833 ØF            rrca
23834 B5            or 1
23835 6F            ld l,a
23836 3E F8         ld a,248
23838 A4            and h
23839 67            ld h,a

ENCONTRAR COMIENZO DE CAR.
EN GENERADOR DE CAR.

2384Ø F1            pop af
23841 E5            push hl
23842 11 ØØ 3C      ld de,1536Ø
23845 6F            ld l,a
23846 26 ØØ         ld h,Ø
23848 29            add hl,hl
23849 29            add hl,hl
2385Ø 29            add hl,hl
23851 19            add hl,de
23852 D1            pop de
```

```
SACAR CAR. A PANTALLA (CON POKE)

23853 Ø6 Ø8             ld b,8
L1
23855 7E                ld a,(hl)
23856 12                ld (de),a


23857 23                inc hl
23858 14                inc d
23859 1Ø FA             djnz L1
```

```
ACTUALIZAR POSICION DE PANTALLA

23861 E1                pop hl
23862 2C                inc l
23863 CB 6D             bit 5,l
23865 28 Ø3             jr z,L2
23867 CB AD             res 5,l
23869 24                inc h
L2
23870 22 BØ 5C          ld (23728),hl
23873 D1                pop de
23874 C9                ret
```

Programa A1.3

```
org 2376Ø

PROGRAMA PARA REEMPLAZAR
SCREEN$ REQUIERE M=LINEA,
L=COLUMNA
VUELVE CON EL REGISTRO A
CONTENIENDO EL CODIGO CAR.
O Ø SI CAR. NO SE ENCUENTRA

2376Ø 26 ØA             ld h,1Ø
23762 2E Ø7             ld l,7
23764 CD D8 5C          call SCR$
23767 C9                ret

RUTINA SCREEN$

SCR$

ENCONTRAR DIRECCION DE PANTALLA
DE CAR.

23768 11 ØØ 4Ø          ld de,16384


23771 19                add hl,de
23772 7C                ld a,h
23773 E6 Ø7             and 7
23775 ØF                rrca
23776 ØF                rrca
23777 ØF                rrca
23778 B5                or l
23779 6F                ld l,a
2378Ø 3E F8             ld a,248
23782 A4                and h
23783 67                ld h,a

EXAMINAR CAR. CON GENE. DE CAR.

23784 11 ØØ 3D          ld de,15616
L3
23787 Ø6 Ø8             ld b,8
23789 ØE ØØ             ld c,Ø
```

```
EXAMINAR SI DE=16384
23791 CB 72             bit 6,d
23793 2Ø 11             jr nz,NFOUND


23795 E5                push hl
L2
23796 1A                ld a,(de)
23797 BE                cp (hl)
23798 2Ø Ø1             jr nz,L1
238ØØ ØC                inc c
L1
238Ø1 13                inc de
238Ø2 24                inc h
238Ø3 1Ø F7             djnz L2
238Ø5 E1                pop hl
238Ø6 CB 59             bit 3,c
238Ø8 2Ø Ø4             jr nz, ENCONTRAR
2381Ø 18 E7             jr L3
NENCONTRAR
23812 AF                xor a
23813 C9                ret


CONVERTIR A CODIGO CAR.


ENCONTRAR


23814 B7                or a
23815 21 ØØ 3C          ld hl,1536Ø
23818 EB                ex de,hl
23819 ED 52             sbc hl,de
23821 Ø6 Ø3             ld b,3
L4
23823 CB 1C             rr h
23825 CB 1D             rr l
23827 1Ø FA             djnz L4
23829 7D                ld a,l
2383Ø 3D                dec a
23831 C9                ret
```

Programa A1.4

```
org 23760
PROGRAMA PARA REEMPLAZAR
PLOT de ROM
REQUIERE H=y, L=x

23760 26 00          ld h,0
23762 2E 5F          ld l,95
23764 CD D8 5C       call PLOT
23767 C9             ret
PLOT


23768 CD DE 5C       call FIND
23771 B6             or (hl)
23772 77             ld (hl),a
23773 C9             ret

ENCONTRAR BIT DE PANTALLA
VOLVER CON HL=BYTE, A=BIT

ENCONTRAR

23774 3E AF          ld a,175
23776 94             sub h

23777 67             ld h,a
23778 11 00 40       ld de,16384
23781 7D             ld a,l
23782 E6 07          and 7
23784 F5             push af
23785 7D             ld a,l
23786 E6 F8          and 248
23788 1F             rra
23789 1F             rra
23790 1F             rra
23791 5F             ld e,a
23792 7C             ld a,h
23793 E6 38          and 56
23795 17             rla
23796 17             rla
23797 B3             or e
23798 5F             ld e,a
23799 7C             ld a,h
23800 E6 07          and 7
23802 B2             or d
23803 57             ld d,a


23804 7C             ld a,h
23805 E6 C0          and 192
23807 1F             rra
23808 1F             rra
23809 1F             rra
23810 B2             or d
23811 57             ld d,a
23812 F1             pop af
23813 2E 80          ld 1,128
L1
23815 A7             and a
23816 FE 00          cp 0
23818 28 05          jr z,L2
23820 CB 1D          rr l
23822 3D             dec a
23823 18 F6          jr L1
L2
23825 7D             ld a,l
23826 EB             ex de,hl
23827 C9             ret
```

Programa A1.5

```
org 23760
PROGRAMA PARA REEMPLAZAR
PLOT OVER 1 DE ROM
REQUIERE H=y, L=x

23760 26 00          ld h,0
23762 2E 5F          ld l,95
23764 CD D8 5C       call PLOTOVER
23767 C9             ret
PLOTOVER
23768 CD DE 5C       call ENCONTRAR
23771 AE             xor (hl)
23772 77             ld (hl),a
23773 C9             ret

ENCONTRAR BIT PANTALLA
VOLVER CON HL=BYTE, A=BIT

ENCONTRAR
23774 3E AF          ld a,175


23776 94             sub h
23777 67             ld h,a
23778 11 00 40       ld de,16384
23781 7D             ld a,l
23782 E6 07          and 7
23784 F5             push af
23785 7D             ld a,l
23786 E6 F8          and 248
23788 1F             rra
23789 1F             rra
23790 1F             rra
23791 5F             ld e,a
23792 7C             ld a,h
23793 E6 38          and 56
23795 17             rla
23796 17             rla
23797 B3             or e
23798 5F             ld e,a
23799 7C             ld a,h
23800 E6 07          and 7
23802 B2             or d


23803 57             ld d,a
23804 7C             ld a,h
23805 E6 C0          and 192
23807 1F             rra
23808 1F             rra
23809 1F             rra
23810 B2             or d
```

```
23811 57            ld d,a
23812 F1            pop af
23813 2e 80         ld l,128
L1
23815 A7            and a
23816 FE 00         cp 0
23818 28 05         jr z,L2
23820 CB 1D         rr l
23822 3D            dec a
23823 18 F6         jr L1
L2
23825 7D            ld a,l
23826 EB            ex de,hl
23827 C9            ret
```

Programa A1.6

```
org 23760
PROGRAMA PARA
REEMPLAZAR POINT DE ROM
REQUIERE H=y, L=x

23760 26 00         ld h,0
23762 2E 5F         ld l,95
23764 CD D8 5C      call POINT
23767 C9            ret
POINT
23768 CD E0 5C      call ENCONTRAR
23771 A6            and (hl)
23772 C8            ret z
23773 3E 01         ld a,1
23775 C9            ret

ENCONTRAR BIT PANTALLA
VOLVER CON HL=BYTE, A=BIT

ENCONTRAR

23776 3E AF         ld a,175
23778 94            sub h
23779 67            ld h,a
23780 11 00 40      ld de,16384
23783 7D            ld a,l
23784 E6 07         and 7
23786 F5            push af
23787 7D            ld a,l
23788 E6 F8         and 248
23790 1F            rra
23791 1F            rra
23792 1F            rra
23793 5F            ld e,a
23794 7C            ld a,h
23795 E6 38         and 56
23797 17            rla
23798 17            rla
23799 B3            or e
23800 5F            ld e,a
23801 7C            ld a,h
23802 E6 07         and 7

23804 B2            or d
23805 57            ld d,a
23806 7C            ld a,h
23807 E6 C0         and 192
23809 1F            rra
23810 1F            rra
23811 1F            rra
23812 B2            or d
23813 57            ld d,a
23814 F1            pop af
23815 2E 80         ld l,128
L1
23817 A7            and a
23818 FE 00         cp 0
23820 28 05         jr z,L2
23822 CB 1D         rr l
23824 3D            dec a
23825 18 F6         jr L1
L2
23827 7D            ld a,l
23828 EB            ex de,hl

23829 C9            ret
```

Programa A1.7

Apéndice 2

Listados del código máquina de la Spectrum (Z80)

NOTAS: (HL)—el número contenido en la dirección a la que apunta el par de registros HL
NN se ingresa con el segundo número primero, ejemplo, 4000 se ingresa 0040 (en hex.)
Registro A—el acumulador de propósito-general

Decimal	*Bytes*	*Hex*	*Nemotécnico*	*Descripción*
0	1	00	nop	No operación
1	3	01	ld bc,NN	Cargar BC con NN
2	1	02	ld (bc),a	Guardar A en (BC)
3	1	03	inc bc	Incrementar BC en 1
4	1	04	inc b	Incrementar B en 1
5	1	05	dec b	Decrementar B en 1
6	2	06	ld b,N	Cargar B con N
7	1	07	rlca	Rotación circular izquierda de A
8	1	08	ex af, af	Poner en activo primer AF
9	1	09	add hl,bc	BC+HL→HL
10	3	0A	ld a(bc)	Cargar A con número en localización (BC)
11	1	0B	dec bc	Decrementar BC en 1
12	1	0C	inc c	Incrementar C en 1
13	1	0D	dec c	Decrementar C en 1
14	2	0E	ld c,N	Cargar C con N
15	1	0F	rrca	Rotación circular derecha de A
16	2	10	djnz x	Decrementar B y JR si B ≠ 0, + o −x
17	3	11	ld de,NN	Cargar DE con NN
18	1	12	ld(de),a	Guardar A en (DE)
19	1	13	inc de	Incrementar DE en 1
20	1	14	inc d	Incrementar D en 1
21	1	15	dec d	Decrementar D en 1
22	2	16	ld d,N	Cargar D con N
23	1	17	rla	Rotación izquierda de A a través del acarreo
24	2	18	jr x	Salto relativo incondicional, + o −x
25	1	19	add hl,de	DE+HL→HL
26	3	1A	ld a,(de)	Cargar A con localización (DE)
27	1	1B	dec de	Decrementar DE en 1
28	1	1C	inc e	Incrementar E en 1
29	1	1D	dec e	Decrementar E en 1
30	2	1E	ld e,N	Cargar E con N
31	1	1F	rra	Rotar A a derecha a través del acarreo
32	2	20	jr nz,x	Salto relativo si no-cero, + o −x
33	3	21	ld hl,NN	Cargar HL con NN
34	3	22	ld(NN),hl	Guardar HL en localización NN
35	1	23	inc hl	Incrementar HL en 1
36	1	24	inc h	Incrementar H en 1
37	1	25	dec h	Decrementar H en 1
38	2	26	ld h,N	Cargar H con N
39	1	27	daa	Ajuste decimal de A
40	2	28	jr z, x	Salto relativo si cero + o −x
41	1	29	add hl,hl	HL+HL→HL
42	3	2A	ld hl,(NN)	Cargar HL con localización (NN)
43	1	2B	dec hl	Decrementar HL en 1
44	1	2C	inc l	Incrementar L en 1
45	1	2D	dec l	Decrementar L en 1
46	2	2E	ld l,N	Cargar L con N
47	1	2F	cpl	Complementar A (comp. en 1)
48	2	30	jr nc, x	Salto relativo si no acarreo, + o −x
49	3	31	ld sp,NN	Cargar puntero pila con NN
50	3	32	ld(NN),a	Guardar A en localización NN
51	1	33	inc sp	Incrementar SP en 1
52	1	34	inc (hl)	Incrementar (HL) en 1
53	1	35	dec(hl)	Decrementar (HL) en 1
54	2	36	ld (hl),N	Guardar N en (HL)
55	1	37	scf	Poner a 1 bandera de acarreo
56	2	38	jr c, x	Salto relativo si acarreo, + o −x
57	1	39	add hl,sp	SP+HL→HL
58	3	3A	ld a,(NN)	Cargar A con localización (NN)
59	1	3B	dec sp	Decrementar SP en 1
60	1	3C	inc a	Incrementar A en 1
61	1	3D	dec a	Decrementar A en 1

Decimal	*Bytes*	*Hex*	*Nemotécnico*	*Descripción*
62	2	3E	Id a,N	Cargar A con N
63	1	3F	ccf	Complementar bandera de acarreo
64	1	40	Id b,b	Mover B a B
65	1	41	Id b,c	Mover C a B
66	1	42	Id b,d	Mover D a B
67	1	43	Id b,e	Mover E a B
68	1	44	Id b,h	Mover H a B
69	1	45	Id b,l	Mover L a B
70	1	46	Id b,(hl)	Mover (HL) a B
71	1	47	Id b,a	Mover A a B
72	1	48	Id c,b	Mover B a C
73	1	49	Id c,c	Mover C a C
74	1	4A	Id c,d	Mover D a C
75	1	4B	Id c,e	Mover E a C
76	1	4C	Id e,h	Mover H a C
77	1	4D	Id c,l	Mover L a C
78	1	4E	Id c,(hl)	Mover (HL) a C
79	1	4F	Id c,a	Mover A a C
80	1	50	Id d,b	Mover B a D
81	1	51	Id d,c	Mover C a D
82	1	52	Id d,d	Mover D a D
83	1	53	Id d,e	Mover E a D
84	1	54	Id d,h	Mover H a D
85	1	55	Id d,l	Mover L a D
86	1	56	Id d,(hl)	Mover (HL) a D
87	1	57	Id d,a	Mover A a D
88	1	58	Id e,b	Mover B a E
89	1	59	Id e,c	Mover C a E
90	1	5A	Id e,d	Mover D a E
91	1	5B	Id e,e	Mover E a E
92	1	5C	Id e,h	Mover H a E
93	1	5D	Id e,l	Mover L a E
94	1	5E	ld e,(hl)	Mover (HL) a E
95	1	5F	ld e,a	Mover A a E
96	1	60	Id h,b	Mover B a H
97	1	61	Id h,c	Mover C a H
98	1	62	Id h,d	Mover D a H
99	1	63	Id h,e	Mover E a H
100	1	64	Id h,h	Mover H a H
101	1	65	Id h,l	Mover L a H
102	1	66	Id h,(hl)	Mover (HL) a H
103	1	67	Id h,a	Mover A a H
104	1	68	Id l,b	Mover B a L
105	1	69	Id l,c	Mover C a L
106	1	6A	Id l,d	Mover D a L
107	1	6B	Id l,e	Mover E a L
108	1	6C	Id l,h	Mover H a L
109	1	6D	Id l,l	Mover L a L
110	1	6E	Id l,(hl)	Mover (HL) a L
111	1	6F	Id l,a	Mover A a L
112	1	70	Id (hl),b	Mover B a (HL)
113	1	71	Id (hl),c	Mover C a (HL)
114	1	72	Id(hl),d	Mover D a (HL)
115	1	73	Id (hl),e	Mover E a (HL)
116	1	74	Id (hl),h	Mover H a (HL)
117	1	75	Id (hl),l	Mover L a (HL)
118	1	76	halt	PARAR
119	1	77	Id (hl),a	Mover A a (HL)
120	1	78	Id a,b	Mover B a A
121	1	79	Id a,c	Mover C a A
122	1	7A	Id a,d	Mover D a A
123	1	7B	Id a,e	Mover E a A
124	1	7C	Id a,h	Mover H a A
125	1	7D	Id a,l	Mover L a A
126	1	7E	Id a,(hl)	Mover (HL) a A
127	1	7F	Id a,a	Mover A a A
128	1	80	add a,b	B + A → A
129	1	81	add a,c	C + A → A
130	1	82	add a,d	D + A → A
131	1	83	add a,e	E + A → A
132	1	84	add a,h	H + A → A
133	1	85	add a,l	L + A → A
134	1	86	add a,(hl)	(HL) + A → A
135	1	87	add a,a	A + A → A
136	1	88	adc a,b	B + A + acarreo → A
137	1	89	adc a,c	C + A + acarreo → A
138	1	8A	adc a,d	D + A + acarreo → A

Decimal	*Bytes*	*Hex*	*Nemotécnico*	*Descripción*
139	1	8B	adc, a,e	E + A + acarreo → A
140	1	8C	adc, a,h	H + A + acarreo → A
141	1	8D	adc, a,l	L + A + acarreo → A
142	1	8E	adc a,(hl)	(HL) + A + acarreo → A
143	1	8F	adc a,a	A + A + acarreo → A
144	1	90	sub b	A − B → A
145	1	91	sub c	A − C → A
146	1	92	sub d	A − D → A
147	1	93	sub e	A − E → A
148	1	94	sub h	A − H → A
149	1	95	sub l	A − L → A
150	1	96	sub (hl)	A − (HL) → A
151	1	97	sub a	A − A → A
152	1	98	sbc a,b	A − B − acarreo → A
153	1	99	sbc a,c	A − C − acarreo → A
154	1	9A	sbc a,d	A − D − acarreo → A
155	1	9B	sbc a,e	A − E − acarreo → A
156	1	9C	sbc a,h	A − H − acarreo → A
157	1	9D	sbc a,l	A − L − acarreo → A
158	1	9E	sbc a,(hl)	A − (HL) − acarreo → A
159	1	9F	sbc a,a	A − A − acarreo → A
160	1	A0	and b	A y B → A
161	1	A1	and c	A y C → A
162	1	A2	and d	A y D → A
163	1	A3	and e	A y E → A
164	1	A4	and h	A y H → A
165	1	A5	and l	A y L → A
166	1	A6	and (hl)	A y (HL) → A
167	1	A7	and a	A y A → A
168	1	A8	xor b	A exclusivo o B → A
169	1	A9	xor c	A exclusivo o C → A
170	1	AA	xor d	A exclusivo o D → A
171	1	AB	xor e	A exclusivo o E → A
172	1	AC	xor h	A exclusivo o H → A
173	1	AD	xor l	A exclusivo o L → A
174	1	AE	xor (hl)	A exclusivo o (HL) → A
175	1	AF	xor a	A exclusivo o A → A
176	1	B0	or b	A o B → A
177	1	B1	or c	A o C → A
178	1	B2	or d	A o D → A
179	1	B3	or e	A o E → A
180	1	B4	or h	A o H → A
181	1	B5	or l	A o L → A
182	1	B6	or (hl)	A o (HL) → A
183	1	B7	or a	A o A → A
184	1	B8	cp b	Comparar A:B
185	1	B9	cp c	Comparar A:C
186	1	BA	cp d	Comparar A:D
187	1	BB	cp e	Compare A:E
188	1	BC	cp h	Comparar A:H
189	1	BD	cp l	Comparar A:L
190	1	BE	cp (hl)	Comparar A:(HL)
191	1	BF	cp a	Comparar A:A
192	1	C0	ret nz	Volver si no-cero
193	1	C1	pop bc	Sacar BC de la pila
194	3	C2	jp nz, NN	Saltar a NN si no-cero
195	3	C3	jp NN	Salto incondicional a NN
196	3	C4	call nz,NN	Llamada NN si no-cero
197	1	C5	push bc	Poner BC en la pila
198	2	C6	add a,N	A + N → A
199	1	C7	rst 0	Llamada 0000 rutina de inicio
200	1	C8	ret z	Volver si cero
201	1	C9	ret	Volver
202	3	CA	jpz,NN	Saltar a NN si cero
203		CB		Ver grupo especial de rutinas CB
204	3	CC	call z,NN	Llamada NN si cero
205	3	CD	call NN	Llamada NN
206	2	CE	adc a,N	A + N + acarreo → A
207	1	CF	rst 8	Llamada 0008 rutina de error
208	1	D0	ret nc	Volver si acarreo = 0
209	1	D1	pop de	Sacar DE de la pila
210	3	D2	jp nc,NN	Saltar a NN si acarreo = 0
211	2	D3	out (N),a	Sacar A al puerto N
212	3	D4	call nc,NN	Llamada a NN si acarreo = 0

Decimal	*Bytes*	*Hex*	*Nemotécnico*	*Descripción*
213	1	D5	push de	Meter DE en la pila
214	2	D6	sub N	A − N → A
215	1	D7	rst 16	Llamada 0010 rutina de impresión
216	1	D8	ret c	Volver si acarreo = 1
217	1	D9	exx	Poner en activo primer B-F (registros de intercambio)
218	3	DA	jp c,NN	Saltar a NN si acarreo = 1
219	2	DB	in a,(N)	Ingresar A desde el puerto N
22Ø	3	DC	call c,NN	Llamada NN si acarreo = 1
221		DD	prefija instrucciones utilizando iy	Ver conjunto especial de rutinas DD
222	2	DE	sbc a,N	A − N − acarreo → A
223	1	DF	rst 24	Llamada 0018 rutina de carácter (1)
224	1	EØ	ret po	Volver si bandera de "overflow"/paridad = 0
225	1	E1	pop hl	Sacar HL de la pila
226	3	E2	jp po,NN	Saltar a NN si bandera de "overflow"/paridad = 0
227	1	E3	ex (sp),hl	Intercambiar (SP) y HL
228	3	E4	call po,NN	Llamada NN si bandera de "overflow"/paridad =
229	1	E5	push hl	0
23Ø	2	E6	and N	Poner HL en la pila
231	1	E7	rst 32	A y N → A
232	1	E8	ret pe	Llamada 0020 rutina de carácter (2)
233	1	E9	jp (hl)	Volver si bandera de "overflow"/paridad = 1
234	3	EA	jp pe,NN	Saltar a localización (HL)
235	1	EB	ex de,hl	Saltar a NN si bandera de "overflow"/paridad = 1
236	3	EC	call pe,NN	Intercambiar DE y HL
237		ED		Llamada NN si bandera de "overflow"/paridad =
238	2	EE	xor N	1
239	1	EF	rst 4Ø	Ver conjunto especial de rutinas ED
24Ø	1	FØ	ret p	A exclusiva o N → A
241	1	F1	pop af	Llamada 0028 rutina de calculadora
242	3	F2	jp p,NN	Volver si bandera de signo = 0
243	1	F3	di	Sacar AF de la pila
244	3	F4	call p,NN	Saltar a NN si bandera de signo = 0
245	1	F5	push af	Deshabilitar interrupciones
246	2	F6	or N	Llamada NN si bandera de signo = 0
247	1	F7	rst 48	Poner AF en pila
248	1	F8	ret m	A o N = A
249	1	F9	ld sp,hl	Llamada 0030 rutina de espacio de trabajo
25Ø	3	FA	jp m,NN	Volver si bandera de signo = 1
251	1	FB	ei	Mover HL a SP
252	3	FC	call m,NN	Saltar a NN si bandera de signo = 1
253		FD	prefija instrucciones	Habilitar interrupciones
			utilizando iy	Llamada a NN si bandera de signo = 1
254	2	FE	cp N	
255	1	FF	rst 56	Ver conjunto especial de rutinas FD
				Comparar A : N
				Llamada 0038

Rut CB

DD 09	ADD	IX,BC	DD CB d 06	RLC	(IX+d)
DD 19	ADD	IX,DE	DD CB d 0E	RRC	(IX+d)
DD 21 +dddd	LD	IX,+dddd	DD CB d 16	RL	(IX+d)
DD 22 addr	LD	(addr),IX	DD CB d 1E	RR	(IX+d)
DD 23	INC	IX	DD CB d 26	SLA	(IX+d)
DD 29	ADD	IX,IX	DD CB d 2E	SRA	(IX+d)
DD 2A addr	LD	IX,(addr)	DD CB d 3E	SRL	(IX+d)
DD 2B	DEC	IX	DD CB d 46	BIT	0,(IX+d)
DD 34 d	INC	(IX+d)	DD CB d 4E	BIT	1,(IX+d)
DD 35 d	DEC	(IX+d)	DD CB d 56	BIT	2,(IX+d)
DD 36 d +dd	LD	(IX+d),+dd	DD CB d 5E	BIT	3,(IX+d)
DD 39	ADD	IX,SP	DD CB d 66	BIT	4,(IX+d)
DD 46 d	LD	B,(IX+d)	DD CB d 6E	BIT	5,(IX+d)
DD 4E d	LD	C,(IX+d)	DD CB d 76	BIT	6,(IX+d)
DD 56 d	LD	D,(IX+d)	DD CB d 7E	BIT	7,(IX+d)
DD 5E d	LD	E,(IX+d)	DD CB d 86	RES	0,(IX+d)
DD 66 d	LD	H,(IX+d)	DD CB d 8E	RES	1,(IX+d)
DD 6E d	LD	L,(IX+d)	DD CB d 96	RES	2,(IX+d)
DD 70 d	LD	(IX+d),B	DD CB d 9E	RS	3,(IX+d)
DD 71 d	LD	(IX+d),C	DD CB d A6	RES	4,(IX+d)
DD 72 d	LD	(IX+d),D	DD CB d AE	RES	5,(IX+d)
DD 73 d	LD	(IX+d),E	DD CB d B6	RES	6,(IX+d)
DD 74 d	LD	(IX+d),H	DD CB d BE	RES	7,(IX+d)
DD 75 d	LD	(IX+d),L	DD CB d C6	SET	0,(IX+d)
DD 77 d	LD	(IX+d),A	DD CB d CE	SET	1,(IX+d)
DD 7E d	LD	A,(IX+d)	DD CB d D6	SET	2,(IX+d)
	ADD	A,(IX+d)	DD CB d DE	SET	3,(IX+d)
DD 8E d	ADC	A,(IX+d)	DD CB d E6	SET	4,(IX+d)
DD 96 d	SUB	(IX+d)	DD CB d EE	SET	5,(IX+d)
DD 9E d	SBC	A,(IX+d)	DD CB d F6	SET	6,(IX+d)
DD A6 d	AND	(IX+d)	DD CB d FE	SET	7,(IX+d)
DD AE d	XOR	(IX+d)	DD E1	POP	IX
DD B6 d	OR	(IX+d)	DD E3	EX	(SP),IX
DD BE d	CP	(IX+d)	DD E5	PUSH	IX
			DD E9	JP	(IX)
			DD F9	LD	SP,IX

Instrucciones ED

ED 40 **IN** **B,(C)**	**ED 50** **IN** **D,(C)**	**ED 60** **IN** **H,(C)**		**ED A0** **LDI**	**ED B0** **LDIR**
ED 41 **OUT** **(C),B**	**ED 51** **OUT** **(C),D**	**ED 61** **OUT** **(C),H**		**ED A1** **CPI**	**ED B1** **CPIR**
ED 42 **SBC** **HL,BC**	**ED 52** **SBC** **HL,DE**	**ED 62** **SBC** **HL,HL**	**ED 72** **SBC** **HL,SP**	**ED A2** **INI**	**ED B2** **INIR**
ED 43 **(addr),BC**	**ED 53** **(addr),DE**	**ED 63** **(addr),HL**	**ED 73** **(addr),SP**	**ED A3** **OUTI**	**ED B3** **OTIR**
ED 44 **NEG**					
ED 45 **RETN**					
ED 46 **IM 0**	**ED 56** **IM 1**	**ED 66** **IM 2**			
ED 47 **LD** **I,A**	**ED 57** **LD** **A,I**	**ED 67** **RRD**			
ED 48 **IN** **C,(C)**	**ED 58** **IN** **E,(C)**	**ED 68** **IN** **L,(C)**	**ED 78** **IN** **A,(C)**	**ED A8** **LDD**	**ED B8** **LDDR**
ED 49 **OUT** **(C),C**	**ED 59** **OUT** **(C),E**	**ED 69** **OUT** **(C),L**	**ED 79** **OUT** **(C),A**	**ED A9** **CPD**	**ED B9** **CPDR**
ED 4A **ADC** **HL,BC**	**ED 5A** **ADC** **HL,DE**	**ED 6A** **ADC** **HL,HL**	**ED 7A** **ADC** **HL,SP**	**ED AA** **IND**	**ED BA** **INDR**
ED 4B **LD** **BC,(addr)**	**ED 5B** **LD** **DE,(addr)**	**ED 6B** **LD** **HL,(addr)**	**ED 7B** **LD** **SP,(addr)**	**ED AB** **OUTD**	**ED BB** **OTRD**
ED 4D **RETI**					
ED 4F **LD** **R,A**	**ED 5F** **LD** **A,R**	**ED 6F** **RLD**			

Apéndice 3

Tabla de conversión decimal-hexadecimal

Decimal 0–255 Hexadecimal 00–FF, byte bajo

Dec.	Hex.	Dec.	Hex.	Dec.	Hex.	2's C.	Dec.	Hex.	2's C.
0	00	64	40	128	80	−128	192	C0	−64
1	01	65	41	129	81	−127	193	C1	−63
2	02	66	42	130	82	−126	194	C2	−62
3	03	67	43	131	83	−125	195	C3	−61
4	04	68	44	132	84	−124	196	C4	−60
5	05	69	45	133	85	−123	197	C5	−59
6	06	70	46	134	86	−122	198	C6	−58
7	07	71	47	135	87	−121	199	C7	−57
8	08	72	48	136	88	−120	200	C8	−56
9	09	73	49	137	89	−119	201	C9	−55
10	0A	74	4A	138	8A	−118	202	CA	−54
11	0B	75	4B	139	8B	−117	203	CB	−53
12	0C	76	4C	140	8C	−116	204	CC	−52
13	0D	77	4D	141	8D	−115	205	CD	−51
14	0E	78	4E	142	8E	−114	206	CE	−50
15	0F	79	4F	143	8F	−113	207	CF	−49
16	10	80	50	144	90	−112	208	D0	−48
17	11	81	51	145	91	−111	209	D1	−47
18	12	82	52	146	92	−110	210	D2	−46
19	13	83	53	147	93	−109	211	D3	−45
20	14	84	54	148	94	−108	212	D4	−44
21	15	85	55	149	95	−107	213	D5	−43
22	16	86	56	150	96	−106	214	D6	−42
23	17	87	57	151	97	−105	215	D7	−41
24	18	88	58	152	98	−104	216	D8	−40
25	19	89	59	153	99	−103	217	D9	−39
26	1A	90	5A	154	9A	−102	218	DA	−38
27	1B	91	5B	155	9B	−101	219	DB	−37
28	1C	92	5C	156	9C	−100	220	DC	−36
29	1D	93	5D	157	9D	−99	221	DD	−35
30	1E	94	5E	158	9E	−98	222	DE	−34
31	1F	95	5F	159	9F	−97	223	DF	−33
32	20	96	60	160	A0	−96	224	E0	−32
33	21	97	61	161	A1	−95	225	E1	−31
34	22	98	62	162	A2	−94	226	E2	−30
35	23	99	63	163	A3	−93	227	E3	−29
36	24	100	64	164	A4	−92	228	E4	−28
37	25	101	65	165	A5	−91	229	E5	−27
38	26	102	66	166	A6	−90	230	E6	−26
39	27	103	67	167	A7	−89	231	E7	−25
40	28	104	68	168	A8	−88	232	E8	−24
41	29	105	69	169	A9	−87	233	E9	−23
42	2A	106	6A	170	AA	−86	234	EA	−22
43	2B	107	6B	171	AB	−85	235	EB	−21
44	2C	108	6C	172	AC	−84	236	EC	−20
45	2D	109	6D	173	AD	−83	237	ED	−19
46	2E	110	6E	174	AE	−82	238	EE	−18
47	2F	111	6F	175	AF	−81	239	EF	−17
48	30	112	70	176	B0	−80	240	F0	−16
49	31	113	71	177	B1	−79	241	F1	−15
50	32	114	72	178	B2	−78	242	F2	−14
51	33	115	73	179	B3	−77	243	F3	−13
52	34	116	74	180	B4	−76	244	F4	−12
53	35	117	75	181	B5	−75	245	F5	−11
54	36	118	76	182	B6	−74	246	F6	−10
55	37	119	77	183	B7	−73	247	F7	−9
56	38	120	78	184	B8	−72	248	F8	−8
57	39	121	79	185	B9	−71	249	F9	−7
58	3A	122	7A	186	BA	−70	250	FA	−6
59	3B	123	7B	187	BB	−69	251	FB	−5
60	3C	124	7C	188	BC	−68	252	FC	−4
61	3D	125	7D	189	BD	−67	253	FD	−3
62	3E	126	7E	190	BE	−66	254	FE	−2
63	3F	127	7F	191	BF	−65	255	FF	−1

Decimal 0–65 280 Hexadecimal 00–FF, byte alto

Decimal	Hex.	Decimal	Hex.	Decimal	Hex.	Decimal	Hex.
0	00	16 384	40	32 768	80	49 152	C0
256	01	16 640	41	33 024	81	49 408	C1
512	02	16 896	42	33 280	82	49 664	C2
768	03	17 152	43	33 536	83	49 920	C3
1 024	04	17 408	44	33 792	84	50 176	C4
1 280	05	17 664	45	34 048	85	50 432	C5
1 536	06	17 920	46	34 304	86	50 688	C6
1 792	07	18 176	47	34 560	87	50 944	C7
2 048	08	18 432	48	34 816	88	51 200	C8
2 304	09	18 688	49	35 072	89	51 456	C9
2 560	0A	18 944	4A	35 328	8A	51 712	CA
2 816	0B	19 200	4B	35 584	8B	51 968	CB
3 072	0C	19 456	4C	35 840	8C	52 224	CC
3 328	0D	19 712	4D	36 096	8D	52 480	CD
3 584	0E	19 968	4E	36 352	8E	52 736	CE
3 840	0F	20 224	4F	36 608	8F	52 992	CF
4 096	10	20 480	50	36 864	90	53 248	D0
4 352	11	20 736	51	37 120	91	53 504	D1
4 608	12	20 992	52	37 376	92	53 760	D2
4 864	13	21 248	53	37 632	93	54 016	D3
5 120	14	21 504	54	37 888	94	54 272	D4
5 376	15	21 760	55	38 144	95	54 528	D5
5 632	16	22 016	56	38 400	96	54 784	D6
5 888	17	22 272	57	38 656	97	55 040	D7
6 144	18	22 528	58	38 912	98	55 296	D8
6 400	19	22 784	59	39 168	99	55 552	D9
6 656	1A	23 040	5A	39 424	9A	55 808	DA
6 912	1B	23 296	5B	39 680	9B	56 064	DB
7 168	1C	23 552	5C	39 936	9C	56 320	DC
7 424	1D	23 808	5D	40 192	9D	56 576	DD
7 680	1E	24 064	5E	40 448	9E	56 832	DE
7 936	1F	24 320	5F	40 704	9F	57 088	DF
8 192	20	24 576	60	40 960	A0	57 344	E0
8 448	21	24 832	61	41 216	A1	57 600	E1
8 704	22	25 088	62	41 472	A2	57 856	E2
8 960	23	25 344	63	41 728	A3	58 112	E3
9 216	24	25 600	64	41 984	A4	58 368	E4
9 472	25	25 856	65	42 240	A5	58 624	E5
9 728	26	26 112	66	42 496	A6	58 880	E6
9 984	27	26 368	67	42 752	A7	59 136	E7
10 240	28	26 624	68	43 008	A8	59 392	E8
10 496	29	26 880	69	43 264	A9	59 648	E9
10 752	2A	27 136	6A	43 520	AA	59 904	EA
11 008	2B	27 392	6B	43 776	AB	60 160	EB
11 264	2C	27 648	6C	44 032	AC	60 416	EC
11 520	2D	27 904	6D	44 288	AD	60 672	ED
11 776	2E	28 160	6E	44 544	AE	60 928	EE
12 032	2F	28 416	6F	44 800	AF	61 184	EF
12 288	30	28 672	70	45 056	B0	61 440	F0
12 544	31	28 928	71	45 312	B1	61 696	F1
12 800	32	29 184	72	45 568	B2	61 952	F2
13 056	33	29 440	73	45 824	B3	62 208	F3
13 312	34	29 696	74	46 080	B4	62 464	F4
13 568	35	29 952	75	46 336	B5	62 720	F5
13 824	36	30 208	76	46 592	B6	62 976	F6
14 080	37	30 464	77	46 848	B7	63 232	F7
14 336	38	30 720	78	47 104	B8	63 488	F8
14 592	39	30 976	79	47 360	B9	63 744	F9
14 848	3A	31 232	7A	47 616	BA	64 000	FA
15 104	3B	31 488	7B	47 872	BB	64 256	FB
15 360	3C	31 744	7C	48 128	BC	64 512	FC
15 616	3D	32 000	7D	48 384	BD	64 768	FD
15 872	3E	32 256	7E	48 640	BE	65 024	FE
16 128	3F	32 512	7F	48 896	BF	65 280	FF

Apéndice 4

Literales de calculadora útiles

Literal d	Función	Acción
1	EXCHANGE	Intercambiar los dos valores superiores
3	SUBTRACT	El valor superior es restado del de abajo
4	MULTIPLY	Multiplicar los dos valores superiores
5	DIVIDE	El valor superior es dividido por el de debajo
15	ADD	Sumar los dos valores superiores
31	SIN	Seno del valor superior
32	COS	Coseno del valor superior
33	TAN	Tangente del valor superior
39	INT	Entero del valor superior
40	SQR	Raíz cuadrada del valor superior
49	DUPLICATE	Se duplica el valor superior y se pone en la parte superior de la pila
52	STK	Apilar datos almacenados en forma comprimida (ver Capítulo 7)
56	END	Terminar rutina de calculadora y volver al programa en código máquina
61	RESTACK	Reemplazar valor superior por su forma de coma flotante
160	0	Apilar 0
161	1	Apilar 1
162	$^1/_2$	Apilar $^1/_2$
163	$\pi/2$	Apilar $\pi/2$
164	10	Apilar 10

Apéndice 5

Mapa de memoria de fichero de visualización

Línea		Línea		Línea	
0	4000 4100 4200 4300 4400 4500 4600 4700	8	4700 4800 4900 4A00 4B00 4C00 4D00 4E00	16	5000 5100 5200 5300 5400 5500 5600 5700
1	4020 4120 4220 4320 4420 4520 4620 4720	9	4720 4820 4920 4A20 4B20 4C20 4D20 4E20	17	5020 5120 5220 5320 5420 5520 5620 5720
2	4040 4140 4240 4340 4440 4540 4640 4740	10	4740 4840 4940 4A40 4B40 4C40 4D40 4E40	18	5040 5140 5240 5340 5440 5540 5640 5740
3	4060 4160 4260 4360 4460 4560 4660 4760	11	4760 4860 4960 4A60 4B60 4C60 4D60 4E60	19	5060 5160 5260 5360 5460 5560 5660 5760
4	4080 4180 4280 4380 4480 4580 4680 4780	12	4780 4880 4980 4A80 4B80 4C80 4D80 4E80	20	5080 5180 5280 5380 5480 5580 5680 5780
5	40AO 41AO 42AO 13AO 44AO 45AO 46AO 47AO	13	47AO 48AO 49AO 4AAO 4BAO 4CAO 4DAO 4EAO	21	50AO 51AO 52AO 53AO 54AO 55AO 56AO 57AO
6	40CO 41CO 42CO 43CO 44CO 45CO 46CO 47CO	14	47CO 48CO 49CO 4ACO 4BCO 4CCO 4DCO 4ECO	22	50CO 51CO 52CO 53CO 54CO 55CO 56CO 57CO
7	40EO 41EO 42EO 43EO 44EO 45EO 46EO 47EO	15	47EO 48EO 49EO 4AEO 4BEO 4CEO 4DEO 4EEO	23	50EO 51EO 52EO 53EO 54EO 55EO 56EO 57EO

Columna																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F

Ejemplo: Quinto byte de carácter en línea 13, columna 27 = 4BAO + 1B = 4B BB h

Apéndice 6

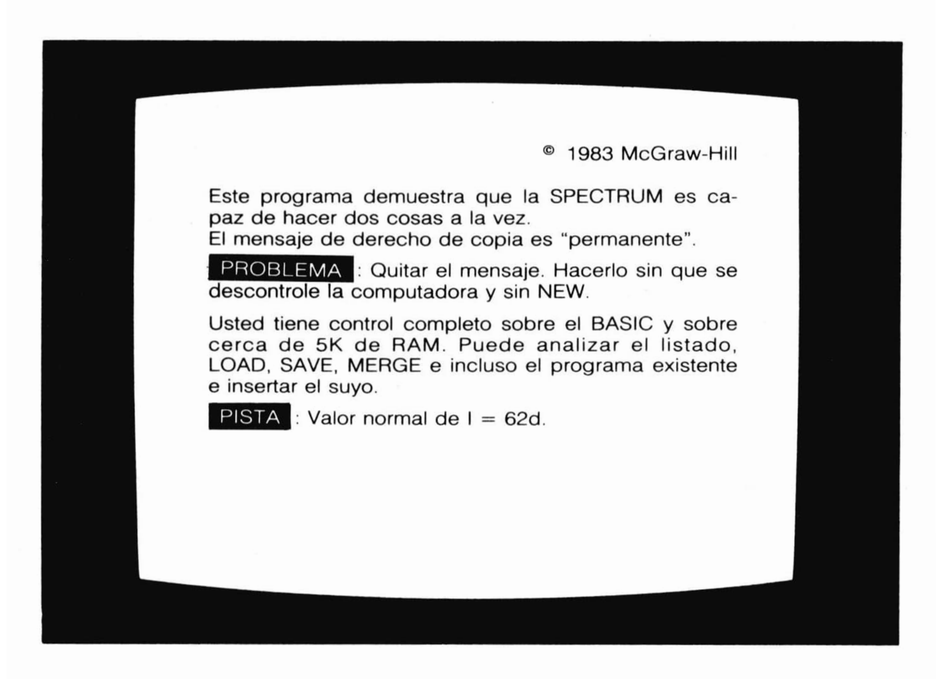
Programa de despedida

Para aquellos de ustedes que quieran desmenuzar programas para ver como funcionan, la cinta contiene un programa en código máquina llamado PROBLEM. El objeto del programa es demostrar el hecho de que la Spectrum puede ejecutar la rutina en código máquina, a la vez que el programador tiene un control completo del BASIC.

Cuando se cargue, la computadora visualizará un mensaje de derecho de copia en la parte superior derecha de la pantalla. Este mensaje de derecho de copia permanecerá constantemente en pantalla. Su problema es averiguar cómo se consigue esto, y cambiar el mensaje sin que se descontrole la Spectrum y sin utilizar NEW.

El programa está pensado de tal forma que sea imposible obtener un listado útil, y tiene varias características que hacen difícil seguir la pista del código máquina.

Si consigue modificar con éxito el mensaje y desensamblar la rutina, verá que le abre una nueva área de posibilidades de programación. Buena suerte.



10 RANDOMIZE USR (PEEK (PEEK 23627+256*PEEK 23628+1) +(256*(PEEK (PEEK 23627+256*PEEK 23628+2))))
20 CLEAR 29100
30 PRINT AT 2,0; "Este programa demuestra que la SPECTRUM es capaz de hacer dos cosas a la vez."' "El mensaje de derecho de copia es 'permanente'"
35 PRINT
40 PRINT "PROBLEMA: Quitar el mensaje."'" "Hacerlo sin que se descontrole la computadora y sin NEW"
50 PRINT
60 PRINT "usted tiene control completo sobre el BASIC y sobre cerca de 5K de RAM. Puede analizar el listado, LOAD, SAVE, MERGE e incluso el programa existente, e insertar el suyo.
70 PRINT '"PISTA : Valor normal de I = 62d"
80 STOP
90 LOAD ""CODE : GO TO 10

Programa "Problem"

OTRAS OBRAS DE INTERES PUBLICADAS POR OSBORNE/McGRAW-HILL

BERMAN: *Explorando el BASIC en el COMMODORE 64.*
CASTLEWITZ: *Introducción al VisiCalc.*
DITLEA: *Guía de Software para microcomputadoras.*
ERICKSON: *Telecomunicaciones con el C-64.*
ERICKSON: *Telecomunicaciones con el MACINTOSH.*
ETTLIN: *Manual de MBASIC.*
ETTLIN: *Introducción al Wordstar.*
FLAST: *1-2-3 RUN. 41 aplicaciones prácticas Lotus 1-2-3.*
FLAST: *54 aplicaciones del VisiCalc.*
FOX: *BASIC básico. Guía para principiantes.*
HEILBORN: *Commodore 64. Guía del usuario.*
HEILBORN: *Programas para ciencias e ingeniería.* Edición Apple II.
HEILBORN: *VIC 20. Guía del usuario.*
HOFFMAN: *Sistema operativo MS-DOS. Guía del usuario.*
HOGAN: *Sistema operativo CP/M. Guía del usuario* (2.ª ed.).
JEFFRIES: *Commodore 64. Pasatiempos y juegos.*
KRUGLINSKY: *Guía a las comunicaciones del IBM/PC.*
MOTTOLA: *Programación en lenguaje ensamblador para el Apple II.*
OSBORNE: *Guía del comprador de sistemas de gestión.*
OSBORNE: *Guía del ordenador personal PET/CBM.*
POOLE: *Algunos programas de uso común en BASIC.*
POOLE: *Algunos programas de uso común en BASIC.* Edición Apple II.
POOLE: *Algunos programas de uso común en BASIC.* Edición Atari.
POOLE: *Algunos programas de uso común en BASIC.* Edición IBM.
POOLE: *Algunos programas de uso común en BASIC.* Edición PET/CBM.
POOLE: *Algunos programas de uso común en BASIC.* Edición TRS-80.
POOLE: *Algunos programas de uso común en Pascal.*
POOLE: *Apple II. Guía del usuario.*
POOLE: *Programas prácticos en BASIC.*
POOLE: *Programas prácticos en BASIC.* Edición Apple II.
POOLE: *Programas prácticos en BASIC.* Edición IBM.
POOLE: *Programas prácticos en BASIC.* Edición TRS-80.
POOLE: *Programas prácticos en Pascal.*
SAND: *Pascal avanzado. Técnicas de programación.*
SACHS: *El IBM/PC.*
STEWART: *Juegos y programas educativos para C-64.*
THOMAS: *Sistema operativo UNIX. Guía del usuario.*
WAITE: *Introducción al procesamiento de palabras.*
TOWNSEND: *Aplique el dBASE II.*
FIELD: *Aplique el MacWhite y MacPaint.*

DISCOGUIAS PUBLICADOS POR OSBORNE/McGRAW-HILL

GIFFOD: *Discoguía para Apple II.*
INGRAHAM: *Discoguía para CP/M.*
JOYANES: *Discoguía para ZX SPECTRUM (TS 2068).*
TAYLOR: *Discoguía para Atari 400/800.*
WILSON: *Discoguía para IBM/PC.*
WILSON: *Discoguía para VisiCalc.*

OTRAS OBRAS DE INTERES PUBLICADAS POR BYTE-BOOKS/McGRAW-HILL

ABELSON: *LOGO para Apple II.*
ABELSON: *Apple Logo.*
BOWLES: *Introducción al UCSD Pascal.*
CIARCIA: *Construya una computadora basada en el Z-80 (Guía de diseño y funcionamiento).*
CURTIS: *WORDSTAR en el IBM/PC.*
DUFF: *Introducción al MACINTOSH.*
GABY: *Gosubs. 100 subrutinas de uso común para el ZX-81 (TS 1000).*
KAMINS: *Usted y la microcomputadora. (Una introducción humanizada a la microinformática.)*
KOLVE: *Guía para seleccionar y adquirir su microcomputador.*
LEWART: *Programas de ciencias e ingeniería para microcomputadoras Sinclair ZX-81 compatibles con el ZX Spectrum.*
MORGAN: *Introducción al microprocesador 8086/8088 (16 bit).*
MULLISH: *Sinclair ZX-81 (TS 1000). Guía del usuario.*
MULLISH: *Applesoft BASIC. Guía para principiantes.*
PECKHAM: *BASIC para Apple II. Manual práctico.*
PECKHAM: *BASIC para Commodore 64. Manual práctico.*
PECKHAM: *BASIC para IBM. Manual práctico.*
PECKHAM: *BASIC para TRS-80 color. Manual práctico.*
SIKONOWIZ: *Introducción al IBM/PC.*
WATT: *Aprendiendo con LOGO.*
SKIER: *Programación en lenguaje ensamblador para VIC-20 y COMMODORE 64.*

OTRAS OBRAS DE INTERES PUBLICADAS POR McGRAW-HILL

ADAMIS: *MACINTOSH. Aplicaciones de MULTIPLAN y MacPaint.*
ADAMIS: *Diccionario BASIC.*
ADAMIS: *Diccionario BASIC del IBM/PC.*
ADAMIS: *Fórmulas y programas usuales en BASIC.*
ADAMIS: *Iniciación al BASIC del IBM/PC.*
BISHOP: *ZX Spectrum (TS 2068). Teoría y proyectos de interfase.*
BUFFINGTON: *Su primera computadora: cómo comprarla y utilizarla.*
GOSLING: *Programación estructurada.*
HURLEY: *Introducción a la programación ZX-81 (TS 1000).*
HURLEY: *ZX Spectrum (TS 2068). Introducción al procesamiento de textos.*
HURLEY: *ZX Spectrum (TS 2068). Programación para jóvenes programadores.*
PHILLIPS: *Programando el Dragón. Juegos y gráficos.*
STREET: *ZX Spectrum (TS 2068). Técnicas de procesamiento de la información.*
WILLIAMS: *ZX Spectrum (TS 2068). Diseño y programación de juegos.*
WOODS: *ZX Spectrum (TS 2068). Programación en lenguaje ensamblador.*
NICHOLLS: *ZX Spectrum (TS 2068). Programación de juegos en lenguaje ensamblador.*

OTRAS OBRAS DE INTERES PUBLICADAS POR McGRAW-HILL SOBRE ZX SPECTRUM (TS 2068)

BISHOP: ZX SPECTRUM (TS 2068). Teoría y proyectos de interfase.
HURLEY: ZX SPECTRUM (TS 2068). Introducción al procesamiento de textos.
HURLEY: ZX SPECTRUM (TS 2068). Programación para jóvenes programadores.
JOYANES: Discoguía para ZX SPECTRUM (TS 2068).
STREET: ZX SPECTRUM (TS 2068). Técnicas de procesamiento de la información.
WILLIAMS: ZX SPECTRUM (TS 2068). Diseño y programación de juegos.
WOODS: ZX SPECTRUM (TS 2068). Programación en lenguaje ensamblador.

ISBN: 968-451-800-5

